DESIGN AND DEVELOPMENT OF AN INTERACTIVE RASTER GRAPHICS TERMINAL

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by
Lt. P. K. DUTT IN

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
SEPTEMBER 1984

EE 1984- M- DU'T. DE S.

5/9/84.

CERTIFICATE

This is to certify that the thesis entitled 'DESIGN AND DEVELOPMENT OF AN INTERACTIVE RASTER GRAPHICS TERMINAL' submitted by Lt. P K Dutt (8210424), for partial fulfilment of the requirements for the award of M.Tech. degree, has been carried out under my supervision. According to my knowledge, it has not been submitted elsewhere for an award of a degree.

5/9/84

Dr. Sanjay K Bose
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology
KANPUR.

September, 1984

ABSTRACT

Interactive Raster Graphics is a powerful medium of man-computer communication. In this project, the hardware has been designed and implemented using microprogrammed finite state machines to implement a 400x300 pixel with 1-bit per pixel interactive raster graphics display. The hardware has been interfaced to an 8085 based Work-Station developed at IIT Kanpur. However, with minor modifications the circuit can be interfaced to any 16-bit processor. The frame buffer memory has been implemented using Dynamic RAM 2118. Microprogrammed control card generates signals to effect a dual port RAM and memory refresh function. Video data at 10 MHz is used to generate images on the CRT. The hardware incorporates Zooming and Panning. Elementary graphics software has been developed in PLM-80 to demonstrate the power of the raster graphics display.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr. Sanjay K. Bose for the guidance and encouragement he gave me all through my project. I shall always be indebted to Mr. Deepak Chaturvedi for having introduced me to hardware design using microprogrammed finite state machines. My friends Mukund and Deshpande were always around to cheer and help me when the going was rough. Mr. C.M. Abraham deserves my thanks for the flawless typing.

The last word of acknowledgement delightedly goes to Paloma who maintained her calm in midst of my often frentic, out-of-control activity.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 ORIGIN OF COMPUTER GRAPHICS

Ever since the computer revolution, there has been a constant search for finding an effective means of communication between man and computers. Computer Graphics, which is a branch of computer technology, provides an effective means of man-computer communication. Here the computer output is displayed in the form of visual images. Computer Graphics owes its popularity to the fact, that the human eye can assimilate the information content of a displayed diagram/ figure much faster than it can scan a table of numbers.

In 1950, the first computer driven display attached to MIT's Whirlwind I computer was developed to generate simple pictures. In 1962, I E Sutherland was the pioneer of the idea of Interactive Computer Graphics. Interactive computer graphics involves two-way communication between the computer and the user. The computer, upon receiving signals from the input device, can modify the displayed picture appropriately. To the user, it appears, that the picture is changing instantaneously in response to his commands. He can give a series of commands, each one generating a graphical response from the computer. In this way he maintains a 'conversation', or dialogue, with the computer.

The main reason for the effectiveness of iteractive computer graphics in various applications is the speed with which the user of the computer can assimilate the displayed information. For example, the engineer designing a circuit layout can see on the screen the various aspects of his layout. With the ability to interact with the computer, the designer can quickly correct any errors and see a revised picture of the same layout. This saves him a lot of trouble and also his performance is greatly enhanced.

In the initial years of development, the popularity of computer graphics was restricted by the expensive hardware and need of substantial computer resources. However, with the steady decrease in cost/performance ratio, computer graphics has found applications in many fields. Some of these are :

a) Tactical warfare simulation
b) Design of logic circuits
c) Animation
d) Engineering design
e) Image processing
f) Printing and plotting
g) Video games
h) Educational purposes
i) Flight simulation.

1.2 AIM OF THIS PROJECT

This project aims at the design and development of an elementary and relatively inexpensive Interactive Raster Scan Graphics Display, and to demonstrate its basic principles of operation.

The graphics hardware design employs a 16-bit internal structure. In the absence of an adequate 16-bit processor system in this department, the graphics display has been developed around an 8-bit processor. However, with a few minor modifications, any 16 bit processor could also be used.

An 8085 μP based 'Work Station', designed and developed at Department of Electrical Engineering, IIT Kanpur, is used to run the graphics software. This work station employs two 8085 processors. One processor is used for keyboard scanning and display refresh and the other for executing the user programs. This results in an uninterrupted execution of the user programs.

Besides this the Workstation has the following features :

a) 6K of user RAM
b) 16K of system program
c) 16K of memory for expansion
d) All pins of 8085 are made available on the edge connector for the user

e) System programs provide an assembler, disassembler, editor, single stepping, trace, break point and many system routines like read, print, etc.

1.3 SUMMARY

The following chapters describe the concepts, architectural design and physical realization of the Interactive Raster Scan display.

- Chapter 2 provides an overview of the various types of graphic displays and explains the principles of a raster graphics display.

- Chapter 3 discusses the effect of CRT specifications on the determination of raster scan display parameters.

- Chapter 4 explains the hardware design problem and then offers a solution for its implementation using microprogrammed finite state machines.

- Chapter 5 provides the hardware implementation details along with timing diagrams.

- Chapter 6 offers suggestion for improving the system performance and scope of further development. A brief explanation of the elementary software developed in PLM-80 to implement graphic functions is also provided.

CHAPTER 2

GRAPHICS DISPLAY - AN OVERVIEW

2.1 ELEMENTS OF A GRAPHICS DISPLAY

The modern graphics display is extremely simple in construction. It consists of three components : a digital memory or frame buffer in which the displayed image is stored as a matrix of intensity values; a television monitor i.e. a home TV set without the tuning and receiving electronics, and a simple interface called display controller, that passes the contents of the frame buffer to the monitor.

Each of these elements determine the quality of the graphics display. The following section discusses each element in brief.

2.1.1 Frame Buffer (Refresh Buffer) :

The frame buffer, also referred to as Refresh Buffer, is a digital memory which stores the displayed image as a matrix of pixels (picture elements). Inside the frame buffer the image is stored as a pattern of binary digital numbers, which represent a rectangular array of pixels. In the most elementary case, where we wish to store only black and white images, we can represent black pixels by 1's in the frame buffer and white pixels by O's .

2.1.2 Display Device :

It is the device on which visual pictures can be displayed. The quality and nature of the displayed picture, to a great extent, is dependent on the following properties of the display device :

a) Aspect ratio
b) Resolution
c) Quality of phosphor
d) Bandwidth
e) Type of video signals accepted i.e. TTL or analog
f) Deflection system.

Although, Cathode Ray Tube (CRT), remains the most popular display device for computer graphics, it suffers the disadvantage of high voltage requirement, bulkiness and weight. Constant research has been in progress to develop a new display device as an alternative to a CRT.

2.1.3 Display Controller (Image Display System) :

The display controller, also referred to as the Display Processing Unit or Image Display System, reads each successive bytes/words of data from the frame buffer and converts O's and 1's into corresponding video signals. These video signals are then fed to the display device to produce, in the simplest case, black and white patterns. This image must be passed to the display device (i.e., the screen must be refreshed), a

certain number of times (usually between 30 to 60 times) per second, in order to maintain a steady flicker free picture on the screen.

In order to change the displayed picture, the contents of the frame buffer can be modified to represent the new pattern of pixels. Fig. 2.1 shows the block diagram of a Graphics Display System.

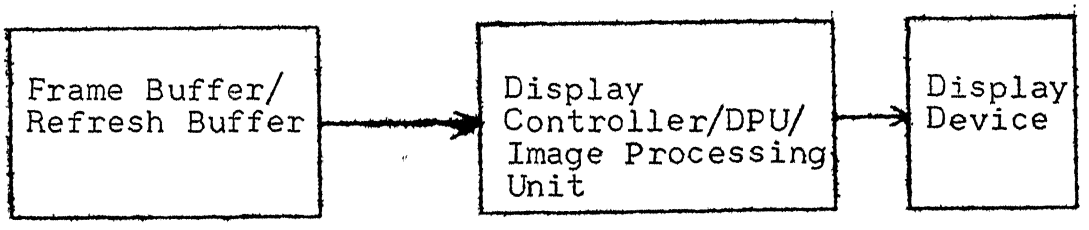


Fig. 2.1 Graphics Display System - Block Diagram

2.2 TYPES OF GRAPHIC DISPLAYS

Computer-generated pictures may be divided broadly into two classes, line drawings and continuous-tone images. Not only are these two classes of images very different in appearance, they require different techniques for their generation. Line drawings are in most respects easier to create for the following reasons :

a. Algorithms for line drawing are simpler

b. Amount of information required to represent them is less.

Continuous-tone images became possible only with the advent of frame buffer displays. The algorithms for generating these

images are still being developed. Although, frame buffer displays draw a line by computing pixel by pixel, which is a very practical way of displaying line drawings, they however, have the following disadvantages.

a. It can not display very smooth lines, since the quantization effect on the screen are always noticeable.

b. It is not suited for highly interactive image manipulations.

In the following sections, the various displays used for line drawing and continuous tone images are briefly discussed.

2.2.1 Storage Tube Display :

This type of display uses a Direct View Storage Tube (DVST) as the display device. A DVST virtually behaves like a CRT with extremely long persistence phosphor. Here, the electron beam is designed not to write directly on the phosphor, but on a fine wire-mesh wire grid, mounted just behind the screen. A pattern of positive charge is deposited on the grid and this pattern is transferred to the phosphor by a continuous flood of electrons issued from a separate 'flood gun'.

In a storage tube display the display controller generates signals, from the computer supplied data, for the DVST. The controller receives a series of instructions from the computer, each specifying a single element of picture. The

controller converts the (x,y) coordinates of the dot to be displayed into voltages that are applied to the deflection yoke to move the beam to the right spot. In fact most of the storage tube displays are designed to plot vectors. The computer supplies the two endpoints of the vector, the display controller positions the beam at the first end point and moves it in a straight path to the other end point. The beam path is determined by a vector generator which feeds the deflection yoke with voltages that change at a steady rate as the vector is being traced out.

The positive charge deposited on the wire-mesh can only be erased by applying positive voltage to the wire mesh for 1 sec. This erase problem prevents the use of the storage tube display for dynamic graphics applications.

2.2.2 Refresh Line-Drawing Display :

This device also contains a controller to convert the computer's output signals into deflection voltages for the yoke of the CRT. Here, the controller is required to operate at high speed to refresh the CRT in order to display a flicker free picture. If the picture contains 5000 vectors, then at say 50 Hz refresh rate, the controller must be able to process 25000 vectors per second. This imposes a severe load on the controller. It is also well beyond the comfortable range of serial asynchronous transmission link. Besides this,

the refresh line-drawing displays are expensive and have a tendency to flicker when the displayed picture is complex.

2.2.3 Raster Scan Display:

In a Raster Scan Display the Frame Buffer (Refresh Buffer) is arranged as a two dimensional array. The entry at a particular row and column stores the brightness and/or color value of the corresponding (x,y) position on the screen in a simple one to one relationship as shown in Fig. 2.2.

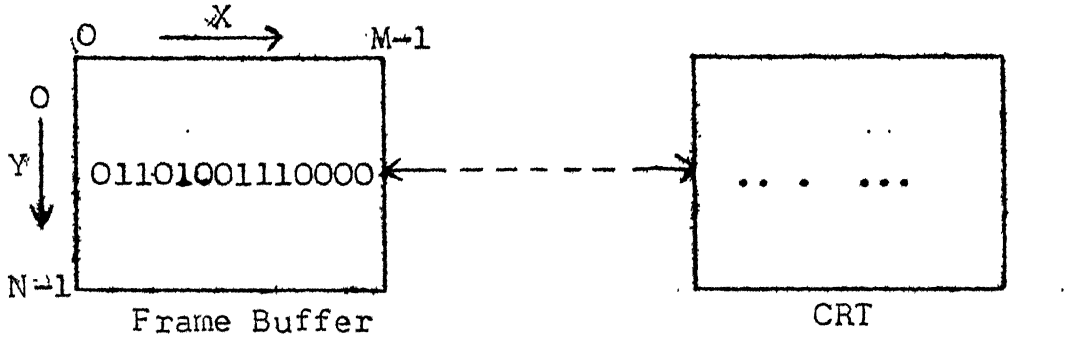


Fig. 2.2 One to one relationship of frame buffer and CRT

Since each memory location defines one point sized element of an image, each screen location (and its corresponding memory location) is often called a pixel or pel (short for picture element). A simple refresh buffer has one bit per pixel and thus defines a two colour (black and white) image.

The display controller cycles through the frame buffer row by row (hence scan line by scan line) typically 30 to 60 times per second. Memory reference addresses are generated

in synchronism with the raster scan, and contents of the memory are used to control the CRT beam's intensity. Figure 2.3 shows the general organization of the display controller.

The raster scan generator produces deflection signals which generate the raster scan, and also controls the X,Y address registers. These registers in turn define the locations of image storage to be displayed in order to control the CRT beam.

At the start of the video refresh cycle, the X and Y registers are set to zero (the top scan line). As the first scan is generated, the X address is incremented up through (M-1). Each pixel value is fetched from the frame buffer and used to control the intensity of the CRT beam. After the first scan line is over, the X address is reset to zero and the Y address is incremented by one. This process continues until the last scan line (y = N-1) is generated. Meanwhile, the computer can make changes in the refresh buffer when permitted by the display controller either during the flyback time or while image (video) refresh is in progress.

The raster scan display overcomes the disadvantages of the earlier displays. Since the scan rate is independent of the complexity of the displayed picture, it can display complex pictures easily. This type of display is also suited

for producing realistic images of solid objects. Nevertheless, the raster display is not without its own set of limitations. In particular, the frame buffer display is not as well suited as the refresh line drawing display, to highly dynamic interactive graphics, since the time taken to fill or change the large amount of frame buffer memory makes interactive response sluggish at times. However, with the availability of, cheap and large capacity random access memory, special purpose integrated circuits raster graphics is fast becoming a very popular method.

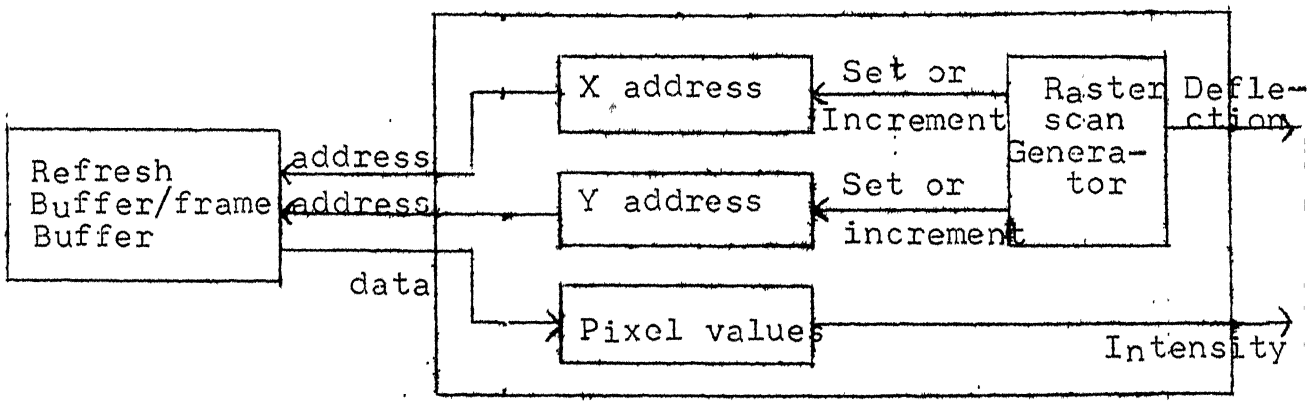


Fig. 2.3 Organization of a Raster Scan Display Controller

CHAPTER 3

CRT SPECIFICATIONS

3.1 INTRODUCTION -

The design of a raster scan graphics display is greatly influenced by the parameters of the display device - the CRT. The characteristics of the CRT not only determine the actual hardware requirement but also the nature and quality of the displayed image. The CRT characteristics are used to determine the following :

a. Number of visible pixels per scan line
b. Number of scan lines per frame
c. Pixel clock rate
d. Frame buffer size
e. Colour and intensity of the displayed image.

The CRT used in this project has the following physical and electrical characteristics :

Size	: 12" diagonal
Phosphor	: P31
Display size	: 8" (H) x 5.75" (V) Nominal
Video	: TTL positive going pulse
Horizontal	: TTL positive going pulse. Pulse width-22 to 40 µsec at 15.75 kHz; Scan frequency 15.750 kHz ± 0.5 kHz
Vertical	: TTL negative going pulse Pulse width 5 to 1400 µsec.

Further details of the CRT specifications along with the timing chart is given in Appendix A to this report.

3.2 DETERMINATION OF RASTER SCAN DISPLAY PARAMETERS

The number of visible pixels and the number of visible scan lines of a raster scan display are influenced by the actual sieze of the CRT. The Aspect Ratio of CRT, used for these calculations, is defined as the ratio of width to the height of the CRT . For commercial graphic displays CRT with an aspect ratio of 2:1 or 1:1 is normally used. However, in the case of the CRT used in this project

$$\text{Aspect Ratio} = \frac{8''}{5.75''} = 4/3$$

This ratio enables us to make efficient use of the display area without restoring to nonuniform scaling in the x and y axes in order to map the frame buffer on to the CRT screen.

3.2.1 Scan lines per frame :

The number of scan lines per frame is decided by the vertical rate and horizontal rate of the CRT used.

In our case, Vertical rate = 50 Hz

Horizontal rate = 15750 Hz

This implies that the CRT can scan 15750 lines per second and also that the CRT must be refreshed 50 times a second.

Therefore, the maximum number of scan lines possible are :

$$\frac{\text{Horizontal Rate}}{\text{Vertical Rate}} = \frac{15750}{50} = 315 \text{ lines.}$$

The number of visible scan lines is affected by the duration of the vertical retrace time. If we assume vertical retrace (and therefore vertical blanking) time of 15 scan line time then

Vertical active time = 315-15

= 300 scan line time.

3.2.2 Pixels per scan line :

In order to maintain uniform scaling in the x and y axes proper care is needed. For this one should take into consideration the aspect ratio while deciding the number of pixels per scan line. For uniform scaling, the ratio of visible pixels per scan line to the number of visible scan lines should satisfy the aspect ratio of the CRT.

Therefore, we have

Number of visible pixels per scan line = 300x4/3

= 400 .

Since, one line time (i.e., the horizontal time) is the sum of horizontal active time and horizontal blanking time, therefore, one line time is the sum of visible pixel time and the

number of pixel time allowed for the horizontal blanking.

3.2.3 Pixel clock rate :

This is the rate at which video data is fed to the CRT. Pixel clock rate, also called dot clock rate, is dependent upon the number of pixels per line and the number of lines in a scan.

The clock rate is determined below :

Pixel clock (Hz) = (N+R) X L X F

where

N = number of visible pixels = 400

L = number of horizontal lines per frame = 315
(300 visible lines+15 line times allowed for vertical retrace)

F = Horizontal refresh rate = 50 Hz

R* = Number of pixel clock times allowed for horizontal retrace time = 240

*(This figure is empirically determined and it establisnes the width of margins on the left and right of CRT display)

Therefore, Pixel clock = (400+240)x315x50

= 10.08 MHz

≈ 10 MHz

3.2.4 Frame buffer size :

The size of frame buffer, for raster scan display using bit mapped memory, is decided by the visible pixels per scan line, number of horizontal lines per scan and the number of bits per pixel. In the simplest case of 1 bit per pixel frame buffer size is determined as

Number of frame buffer bits = 300x400x1

= 117.18 K bits.

If we design the display controller to read 16 bits of memory at a time then

$$\text{Frame buffer requirement} = \frac{300 \times 400 \times 1}{16}$$

= 7.32 K words

Sixteen chips of Dynamic RAM 2118 (16Kx1 bit) are required for implementation of frame buffer.

3.2.5 Intensity and Colour of the displayed image :

In this project a simple black and white monitor has been used. This CRT accepts only TTL video pulses and therefore there is no intensity control and only single intensity black and white images can be represented on the screen. The P31 quality of phosphor used in this CRT gives a green image instead of pure white. Therefore, the displayed picture appears green and black.

CHAPTER 4

DESIGN APPROACH

This chapter discusses the criterion that the raster scan graphics hardware is required to fulfill and then offers a solution for the hardware implementation using microprogrammed finite state machines. These machines are also discussed in brief.

4.1 THE PROBLEM

The design of a raster graphics hardware encompasses the design of the frame buffer and the display processing unit. The design aim has been that the hardware should be able to perform various operations at high speed, with low chip count and that the complexity of the circuit should be relatively less. In the following section the problem has been broken down into the small parts. In fact, the modular approach of hardware design, helps in finding a simple and efficient solution.

4.1.1 Frame Buffer Design Requirements

A substantial amount of memory is required to implement the frame buffer. Dynamic Random Access Memory (DRAM) with its advantage of low cost per bit, low power consumption, and high density is most suited for graphics applications. However, in using DRAMs, the refresh, timing, and arbitration overhead

have to be catered for in the design. The DRAM control circuitry has to generate signals such as Row Address Strobe ($\overline{RAS}$) and Column Address Strobe ($\overline{CAS}$), provide refresh cycles and handle arbitration. This adds to the component count and the overhead costs in an actual design and implementation. One row of the DRAM is refreshed when the DRAM clocks in a row address. The refresh circuitry must sequence through all the row addresses (128 in case of DRAM 2118) within 2 ms for the memory data to be retained. This implies that a memory cycle (read, write, or refresh) must be performed at all row addresses within the specified time. For DRAM 2118, fourteen addresses are required to access each of the 16, 384 data bits. To accomplish this, the 14-bit address is multiplexed onto seven address inputs of 2118. The two 7-bit address words are sequentially latched into the 2118 by two TTL level clocks $\overline{RAS}$ and $\overline{CAS}$. For the implementation of a 300x400 pixel system, 7.32 K 16-bit words of memory is needed. Therefore, sixteen chips of 16K x 1 bit DRAM 2118 are necessary to implement the frame buffer with 16-bit internal structure.

4.1.2 Display Processing Unit - Design Requirements

The display processing unit is the heart of the raster graphics display system. The DPU must work in synchronism with the pixel clock to generate memory cycles for displayed video refresh, memory refresh and CPU access to the frame buffer.

In order to refresh the displayed video, the 16-bit data, from the frame buffer, must be read so as to satisfy the pixel rate. This 16-bit data is loaded into two 8-bit parallel in serial out shift registers clocked at the pixel rate of 10 MHz (i.e., 100 ns/bit). Thus, the shift registers need to be reloaded with fresh data, from the frame buffer, every 1600 ns. This implies a memory cycle time of 1600 ns.

During the horizontal active time of every visible scan line duration (i.e., during vertical active time) video refresh cycles are required to be performed so as to provide video signal to the CRT at 10 MHz. On completion of a video refresh cycle the column and row address must be incremented/ reloaded as explained earlier in Sec. 2.2.3.

Meanwhile, the external processor may like to access the frame buffer to perform a read/write operation. The DPU is required to handle arbitration between the CPU, Video refresh, and Memory refresh cycles attempts at accessing the frame buffer. This arbitration process must honour the processor requests within a reasonable amount of time so as not to significantly slow down the processor. In addition, the DPU should complete 128 cycles for memory refresh within every 2 ms.

To sum up, the DPU is required to perform the following operations :

a. Generate control signals for the memory, i.e. $\overline{RAS}$, $\overline{CAS}$

b. Multiplex the 14-bit address onto the 7 address inputs of the 2118

c. Provide video refresh data to control the CRT beam's intensity at 10 MHz

d. Generate addresses for 128 memory read cycles once every 2 ms for memory refresh

e. Effect a dual port DRAM

f. Accept and acknowledge processor requests

g. Generate memory reference addresses in synchronism with the raster scan

h. Handle arbitration for access to the frame buffer.

4.2 A SOLUTION

The DRAM-2118 has a memory cycle time of 320 ns. This speed is far greater than our requirement of a memory cycle of 1600 ns (Sec. 4.1.2). Therefore, we can afford to divide the 1600 ns time interval into two states, each of 800 ns. In each of these 800 ns states a memory cycle (read, write or refresh) for video refresh, processor access, or memory refresh can be easily performed. If during the horizontal active time, every alternate state is used to access the frame buffer for video refresh then the displayed video refresh rate of 10 MHz is maintained. The remaining states in the horizontal active time could then be used either for memory refresh or for honouring the processor requests.

As we have seen in Sec. 3.2.2 and Sec. 3.2.3

One line time = Horizontal Blanking time + Horizontal Active time

Horizontal Active Time = 400 pixels at pixel rate of 100 ns/bit

Horizontal Blanking Time = 240 pixel at pixel rate of 100 ns/bit.

Since one state time = 800 ns

Therefore, one line time = Horizontal Blanking of 30 state times + Horizontal active time of 50 states

= 80 states (of 800 ns each)

These 80 states of one line time may occur either during vertical blanking time or during vertical active time. In either of the case, each state should be appropriately used.

4.2.1 An Algorithm for DPU Design

An algorithm has been developed in a pascal like language to elucidate the sequence of operations that need to be performed by the DPU for handling each state time in a scan time of 315 line time. The algorithm uses the following notations and variables :

1. V-State-Count : It indicates the line count of a scan time
2. H-State-Count : It indicates the state count of a line time
3. Generate-Cycle : A procedure to generate necessary control signals for the operation of a memory cycle

The procedure generate cycle can generate any of these three memory cycle.

a. Video : In this cycle one address of the frame buffer is read and 16-bit data loaded in shift registers for displayed video refresh

b. Refresh : In this cycle one row of the DRAM is refresh by doing a memory read operation

c. CPU : The cycle during which the processor request for a read/write cycle to the memory is executed and completed.

4. CPU-Request : A Boolean variable which is true if processor request is pending.
5. HBLNK : A Boolean variable for horizontal blanking
6. VBLNK : A Boolean variable for vertical blanking.

```
Begin
  Repeat
     V-State-Count: = 0 (*Initialization*)
     While V-State-Count ≤ 14 do (*during vertical retrace time*)
     Begin
     V-BLNK:=0; H-State-Count:=0; (*Initialization)
     while  H-State-Count ≤ 79 do (*for one line time*)
     Begin
         Case  H-State Count is EVEN
               Generate-Cycle (Refresh)
         Case  H-State Count is ODD
         Begin
             if  CPU-Request = True then  Begin
                 Generate-Cycle (CPU);
                 CPU-Request:= FALSE(*Acknowledge to CPU*)
                 end
         end
         H-State-Count:= H-State-Count+1
     end;
     V-State-Count:= V-State-Count+1
   end
   While  V-State-Count ≤ 314 do
   Begin
      VBLNK:= FALSE; H-State-Count: = 0;
      while  V-State-Count ≤ 29  do
      Begin
          H-BLNK:= True;
          Case  H-Cycle-Count  is  EVEN
                Generate-Cycle (Refresh);
          Case  H-Cycle-Count is ODD
                if H-Cycle-Count < 29  then
                Begin
                  if CPU-Request = True  then
                     Generate-Cycle (CPU)
                  else  Generate-Cycle (Refresh)
                end
              else Generate Cycle (Video)
            end
```

```
            H-State-Count:= H-State-Count + 1
         end
      While  H-State-Count ≤ 79 do
         Begin
            Case: H-State Count is EVEN
               Begin
                  if  CPU-Request = TRUE  then  Begin
                      Generate-Cycle (CPU);
                      CPU-Request: = FALSE;     end
                  else
                      Generate-Cycle (Refresh)
                  end
            Case  H-State-Count is ODD
               Begin
                  if  H State-Count ≤  77 then
                      Generate cycle (Video)
                  else
                      if CPU-Request = TRUE then Begin
                         Generate Cycle (CPU)
                         CPU Request:= FALSE;     end
               end
            H-State-Count:= H-State-Count+1;
         end
      H-State-Count:= 0;
      V-State-Count:= V-State-Count+1
   end
forever
end.
```

The actual algorithm to demonstrate the operations is also given in this section.

The following features of the algorithm need to be highlighted.

a. For memory refresh atleast 40 cycles are completed during each line in the vertical retrace time. At 50 Hz vertical rate, 600 refresh cycles are undertaken in 20ms.

b. Processor request is delayed, at most by 24 µsec. and nearly every 1.6 µsec. processor access is allowed. Thus the processor can work at its maximum speed.

c. Video cycles are allowed every 1.6 µsec. Thus 10 MHz data rate is maintained.

d. The first video cycle occurs during the last state of horizontal blanking time. This ensures that the video signals are available immediately after blanking is removed.

e. The last video cycle is performed at state count of 77. After these 16-bits have been shifted, the blanking is applied.

4.3 MICROPROGRAMMED FINITE STATE MACHINES - BASICS

In a finite state machine the next state is determined uniquely by the present state of the machine and the present

input. Basically, a microprogrammed machine is one in which a coherent sequence of microinstructions is used to execute various operations required by the machine. All the little elemental tasks performed by the machine in executing the machine instruction are called microinstructions. The storage area for these microinstructions is usually called the microprogram memory.

Microprogrammed machines are usually distinguished from nonmicroprogrammed machines in the following manner. Older, nonmicroprogrammed machines implemented the control function by using a combination of gates and flip-flops connected in somewhat random fashion in order to generate the required timing and control signals for the machine. Microprogrammed machines, on the other hand, are normally considered highly ordered, flexible and organized with regard to the control function field. In its simplest definition, a microprogram control unit consists of the microprogram memory and the structure required to determine the address of the next instruction.

The microprogram memory is simply an N-word by M-bit memory used to hold the various microinstructions. Fig. 4.1 depicts the word number definition of an N-word microprogrammed memory. For an N-word memory, the address locations are defined as location Ø through location N-1. For example, a 2K-word microprogram memory will have address location $\emptyset_D$ through 2047_D.

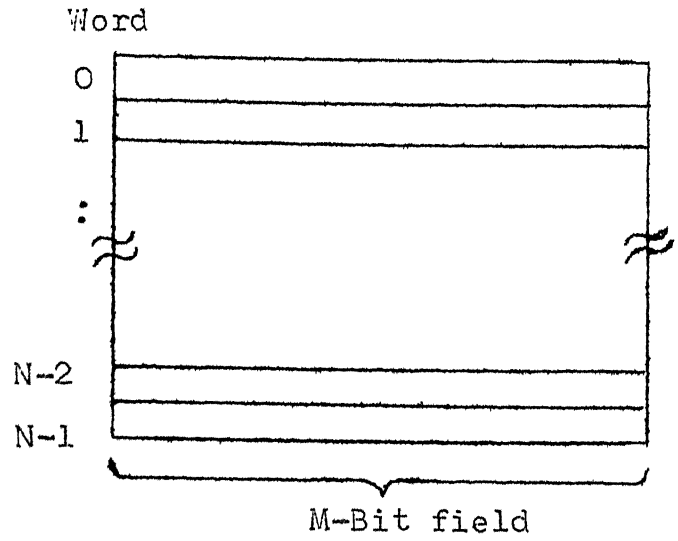


Fig. 4.1 Organization of a N-Word by M-bit Microprogram Memory

Each word of microprogram memory consists of M-bits. These M-bits are usually broken into various field definitions. The field can consist of various number of bits from 1 to M. It is the definition of various fields of a microprogram word that is usually referred to as 'formatting'.

Once the microprogram format has been defined, it is necessary to execute sequences of these microinstructions if the machine is to perform any real function. In its simplest form, all that is required to sequence through a series of microinstructions is a microprogram address counter. Such a simplified microprogram memory address control unit is depicted in Fig. 4.2.

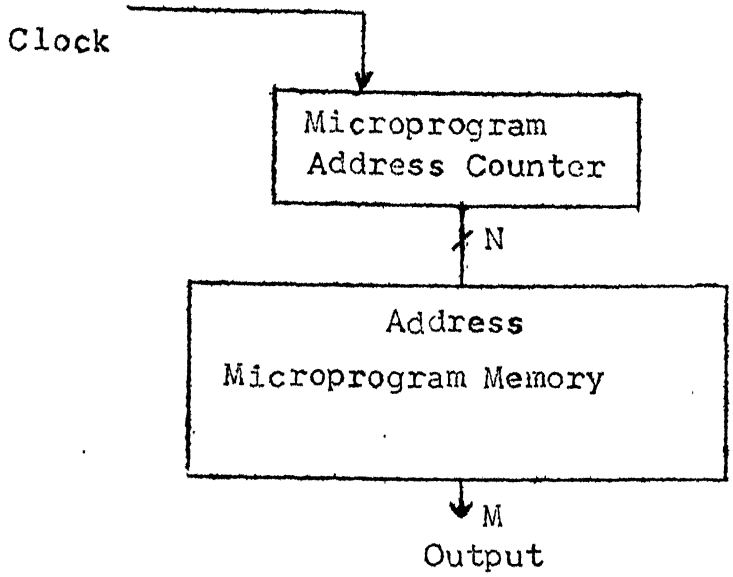


Fig. 4.2 Microprogram Memory Address Control Using a Counter

The microprogram address counter simply increments by one on each clock cycle to select the address of the next instruction.

If the microprogram control unit is to have ability to select other than the next microinstruction, the control unit must be able to load a Branch or Jump address. Fig. 4.3 shows a control unit architecture whereby a branch address can be parallel loaded into the microprogram address control.

The load control is a single bit field within the microprogram word format. Let us call this one-bit field the microprogram address counter load enable bit. When this bit is at logic 1 a load will be inhibited and when this bit is logic 0, a load will be enabled. If the load is enabled the branch address contained in the microprogram memory will be parallel

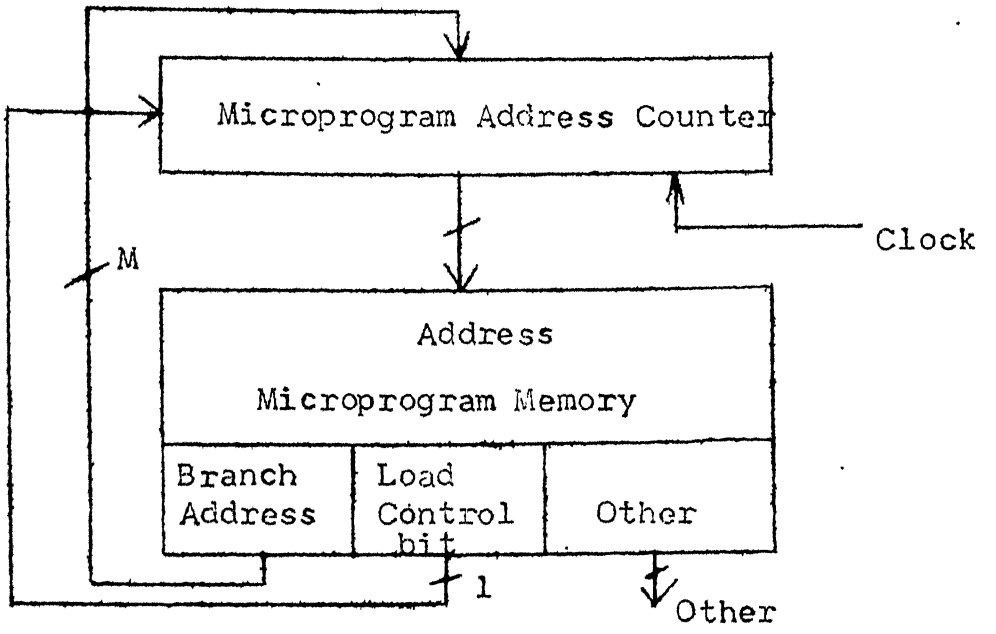


Fig. 4.3 Branching or Jumping in a Microprogram Memory Space requires a branch address and a load control signal

loaded into the microprogram address counter. This results in the ability to perform an N-way branch. This simple branching feature allows a microprogram memory controller to execute sequential microinstruction or Branch or Jump to any address either before or after the address currently contained in the microprogram address counter.

The N-bit address to the microprogram memory can also be a combination of the microprogram address counter outputs and certain conditions/inputs. These conditions/inputs should be stable for each clock time or its multiples. This can be achieved by latching the inputs/conditions at the appropriate

time. The Branch address can be provided by the microprogram memory, as discussed, or in case a fixed Branch is desired, the memory address counter input can be hardwired to the desired value so as to achieve the same effect. Fig. 4.4 depicts a fixed Branch/Jump architecture.

4.4 DESIGN IMPLEMENTATION - (BLOCK DIAGRAM)

The hardware design of the circuit to implement the algorithm discussed in Sec. 4.2.1 is achieved by using three microprogrammed finite state machines. The three machines implemented are :

(a) Macro State Machine (MSM)
(b) Horizontal State Machine (HSM)
(c) Vertical State Machine (VSM)

A central timing unit based on 10 MHz crystal reference controls the activity of the entire circuit. Hardware design using state machines minimises the need for combinatorial logic and permits easy modifications to the circuit in case the system needs to be interfaced to another CRT or changes are to be made in the pixels per line or the number of scan lines per frame.

Fig. 4.5 shows the block diagram of the hardware implementation of DPU and Frame Buffer. The MSM, clocked by the pixel clock of 10 MHz, sequences through eight macro states mS_0 thru mS_7, thereby dividing one state time of 800 ns into

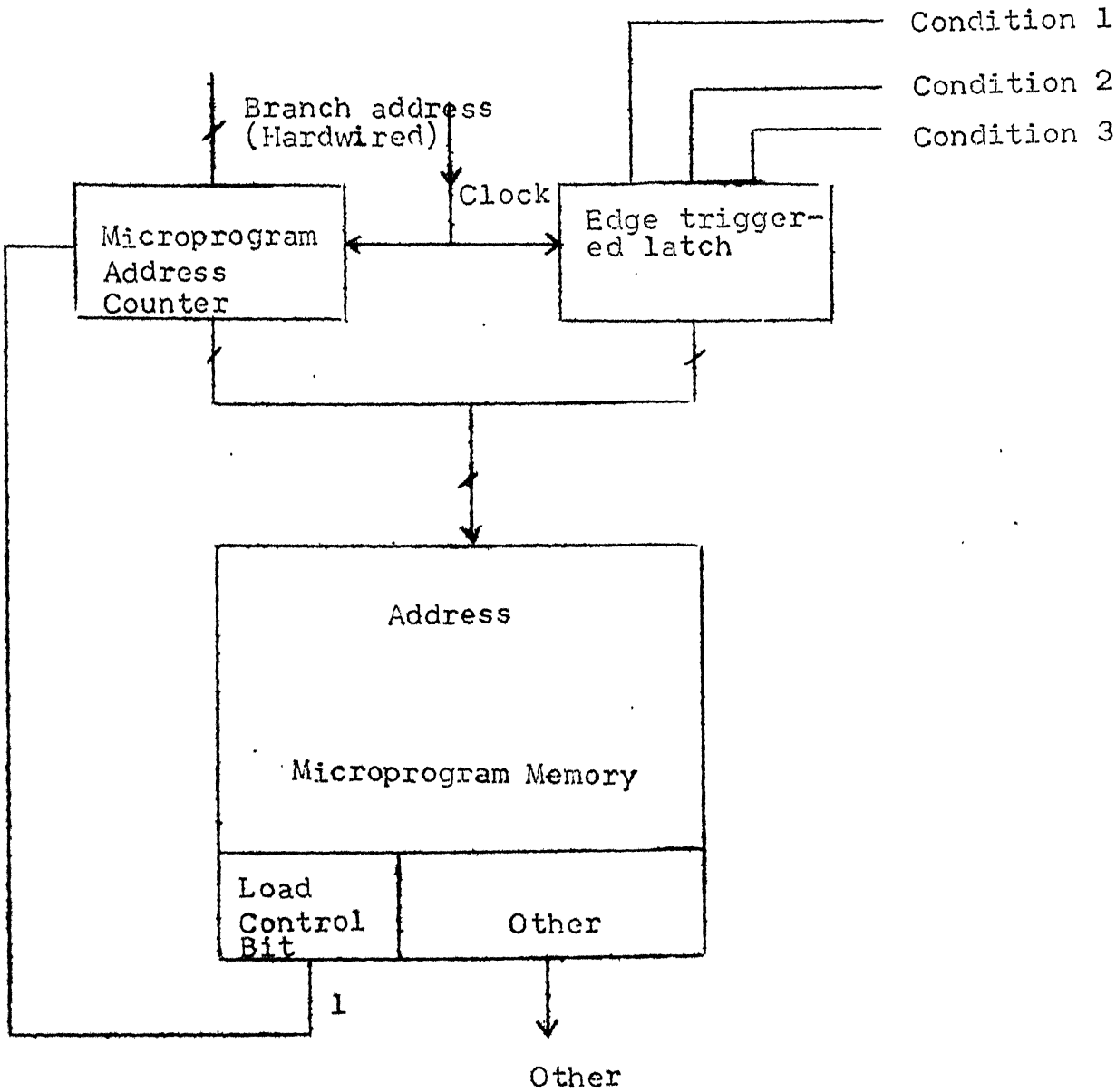


Fig. 4.4 Fixed Branch/Jump architecture

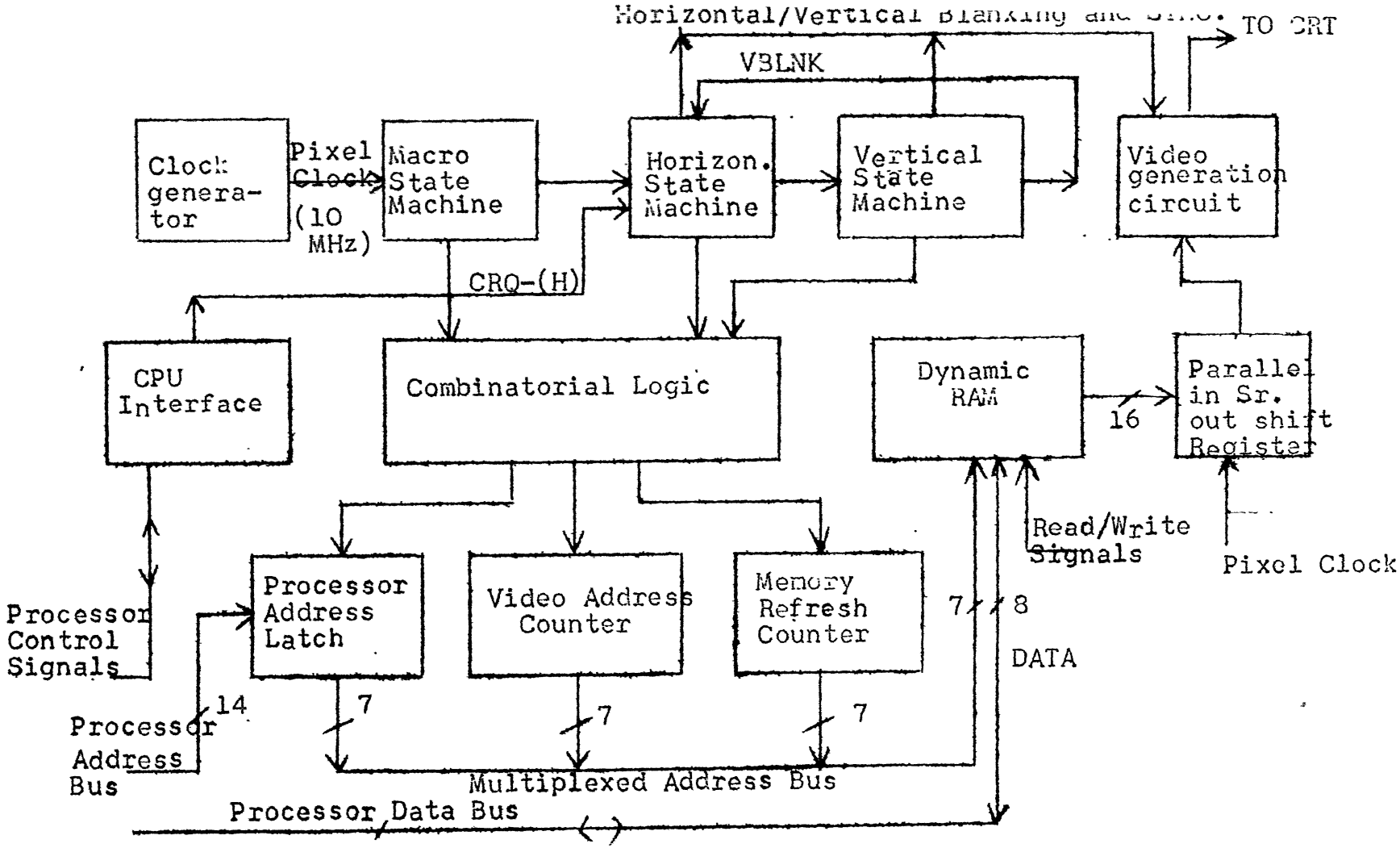


Fig. 4.5 Block diagram of display processing unit and frame buffer

eight equal ports of 100 ms each. This machine generates timing signals for one memory cycle and also produces increment pulse and latch pulse during ms_1 and ms_7 respectively. These pulses are used by the remaining circuit to carry out appropriate operation during the beginning or end of a state. Moreover, the MSM generates timing signal required to divide the one state time into two parts to enable the 14 bit address multiplexing onto a 7-bit address bus.

The HSM is the heart of the system. This machine sequences through 80 states HS_0 through HS_{79}. This machine accepts processor requests for access to the frame buffer and the vertical blanking signal from VSM to handle the problem of arbitration. The HSM schedules the access to the frame buffer guaranteeing timely access by video refresh and memory refresh circuitry interleaved with external processor requests. Since the frame buffer refreshes the display from a dual port RAM the prodessor bus is used only when the workstation desires to update or read the RAM. Thus the processor bus is free for whatever the application may require. The VSM sequences through 315 states VS_0 through VS_{314}. This machine is clocked by the HSM in HS_{79}. Each state of this machine is equivalent to one line time.

The HSM and VSM generate appropriate signals for advance and control of video refresh address counter, memory refresh address counter and processor address buffer. The Frame

Buffer is composed of one RAM array containing 32K bytes of dynamic RAM. This RAM is accessed as 16 bit word for the displayed video refresh and as an 8-bit byte for processor access. Video generation circuit combines the data loaded into the shift register with the blanking and synchronization signals from the HSM and VSM to produce the video signals needed to drive the display.

CHAPTER 5

HARDWARE IMPLEMENTATION

5.1 DESIGN FEATURES

The design problem of a low cost raster graphics hardware is considerably simplified by using a modular design approach employing microprogrammed finite state machines. Most of the components used in this project were available locally. However, in order to reduce the chip count, certain IC's had to be procured from external sources.

The hardware is implemented on two printed circuit boards. These are :

a. Display controller card

b. Frame buffer card.

The display controller card comprises of the three finite state machines and the combinatorial logic required to generate buffered signals for processor read/write cycle, video refresh cycle, memory refresh cycle and CRT synchronization and blanking. The frame buffer card incorporates the DRAM, video refresh address counters, memory refresh address counters, processor interface, Address and Data buffers, Zoom control logic and the Memory bank select logic.

5.2 IMPLEMENTATION DETAILS

A detailed schematic of the entire circuit and the IC layout is given in Appendix C of this report. IC's U1 to U26 are on the Control Card and IC's U27 to U75 are on the Frame Buffer Card. The edge connector pin details for the two cards are also given in Appendix C. For ease of explanation, we shall consider each functional block of the schematic individually.

5.2.1 Macro State Machine

The Macro State Machine (MSM) generates signals for one memory cycle. This machine is constructed using a 4-bit synchronous microprogram address counter U6 (74S161), high speed 32x8 bipolar microprogram memory U10 (82S123) having an access time of 25 ns and an 8-bit,D-type positive edge triggered latch U14 (74LS374). The pixel clock oscillator provides a 10 MHz buffered clock. The pixel clock is used to sequence the MSM through its eight states mS_0 thru mS_7 repetitively. The positive edge of the pixel clock is used to clock the state counter and also to strobe the microprogram memory data into the latch. The same edge of the clock can be used for these two functions as the data hold time for 74LS374 is zero. Thus, the data strobed at time (t) is actually the contents of the memory whose address was generated by the state counter at time (t-1). Since, a time duration of 100 ns

is available before the MSM signals are required to be stable, there is no critical access time.

The format of the control signals generated by the MSM is given below.

a. RAS-(L) : Row address strobe, active low

b. MUX-(H) : Memory address bus multiplexing control

c. CAS-(L) : Column address strobe, active low

d. WE-(L) : Memory write, active low

e. INC-(H) : Increment pulse, active high

f. LATCH-(H):Latch pulse, active high

g. INC-(L) : Increment pulse, active low

h. LATCH-(L):Latch pulse, active low.

The timing relationship of these signals is shown in Fig. 5.1. The microprogram memory address and it's contents in HEX is given in Appendix B.

5.2.2 Horizontal State Machine

The horizontal state machine (HSM) comprises of two 4-bit microprogram address counters U2 and U3 (74LS163), a 2Kx8 microprogram memory U7 (EPROM-2716),a 8-bit,D-type positive edge triggered latch U11 (74LS374) and a 6-bit D-type edge triggered latch U12 (74174).

The microprogram address counter is clocked on the positive edge of every INC-(H) (i.e. every 800 ns) and the

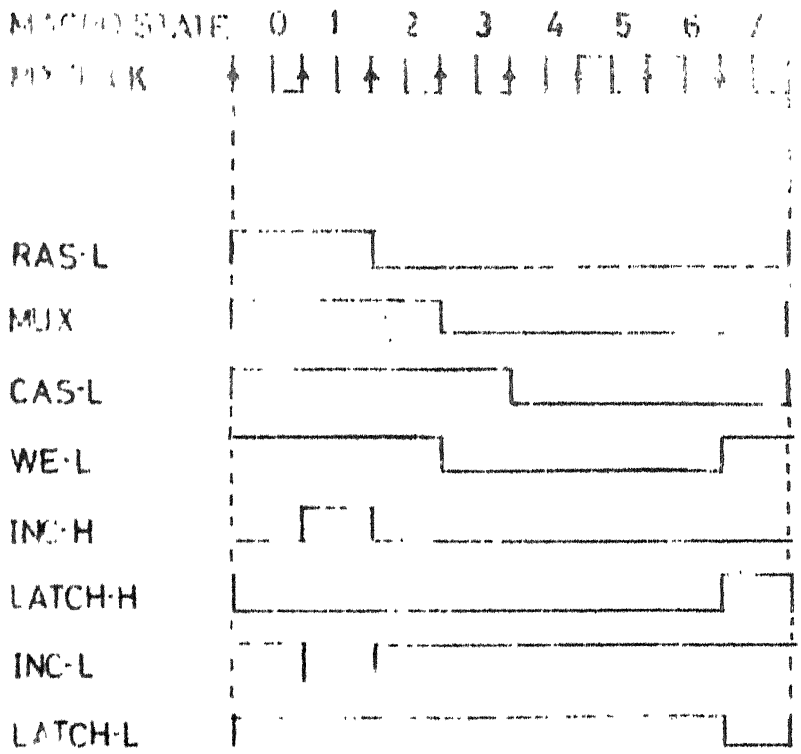


FIG 54 TIMING DIAGRAM OF MSM

microprogram memory data is strobed into the latch U11 on the positive edge of every LATCH-(L). The input conditions, i.e., the processor request status and the zoom factor, are strobed into U12 with INC-(L). The strobed input condition and the counter output decide the memory location to be accessed. Nearly 600 n s are allowed after the address set-up for the data output of EPROM 2716 to stabilize before it is strobed into the HSM output latch U11. Thus, there is no critical access time problem in this state machine.

The HSM microprogram address counters provide a 7-bit address to the microprogram memory. The counter sequences from $\emptyset_D$ to 79_D, to generate the required 80 states $HS_\emptyset$ thru HS_{79}. On completion of the 80 states the address counter is reset to zero by the IVSM-(L) signal applied to the $\overline{Clr}$ input of the counters. In order to overcome any access time problem for this machine the microprogram memory is programmed such that signals required at the output in state N are micro-programmed at microprogram memory address location N-1.

The timing diagram of the various signals at the HSM output is shown in Fig. 5.2. Fig. 5.2(a) shows the relation-ship of the signals generated for one horizontal line time during vertical active time i.e. VBLNK(L)= 1 and Fig. 5.2(b) exhibits the relationship when VBLNK(L)=0(i.e. during vertical blanking time). The input condition VBLNK-(L) is provided by the VSM and therefore controls the nature of the various

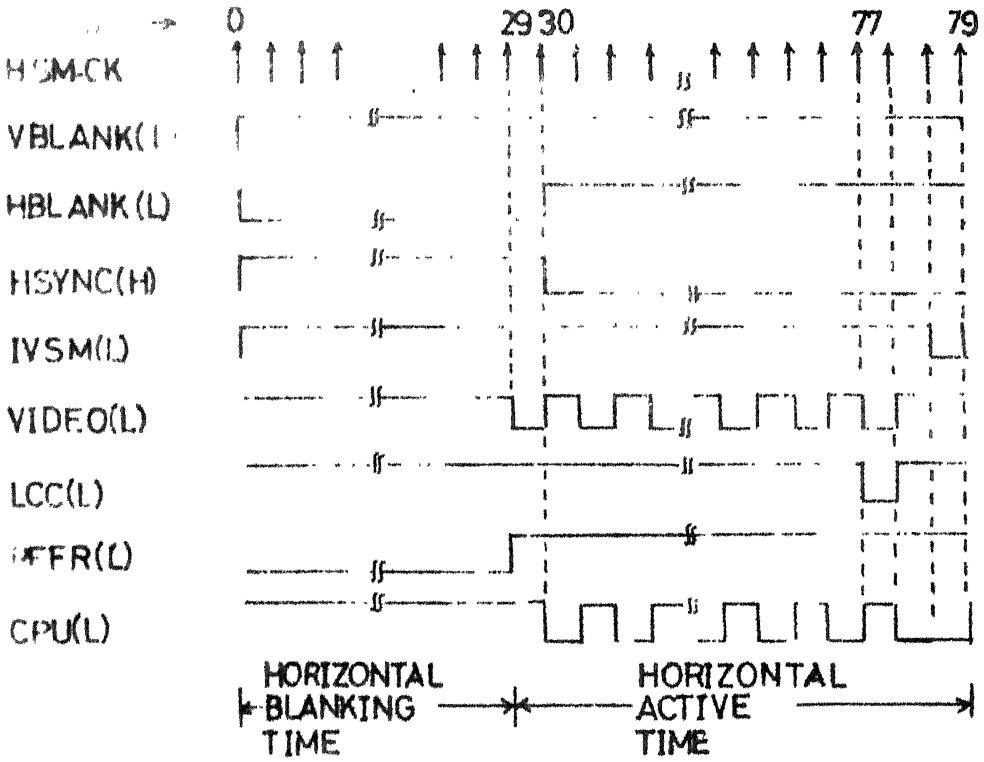


FIG 5·2a TIMING DIAGRAM OF HSM
FOR INPUT CONDITION
VBLANK-L='1' CPU-L='1'

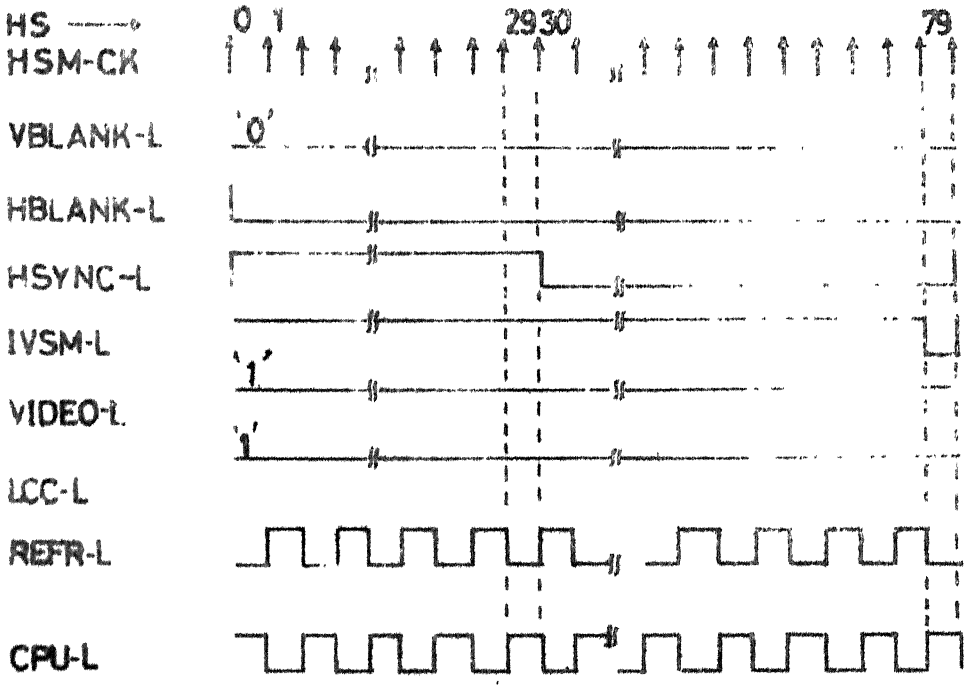


FIG 5-2 b TIMING DIAGRAM OF HSM FOR INPUT CONDITION VBLANK-L='0' CPU-L ='1'

cycles occurring during a line time of 80 states. The processor request circuit provides CRQ-(H) for requesting a CPU cycle. The rate at which the video refresh cycles are generated is decided by the value of zoom control bits ZFØ and ZF1 provided by the Video Refresh Circuit. The format of the signals generated by the HSM is given below :

a. HBLNK-(L) : Horizontal blanking, active low
b. HSYNC(H) : Horizontal synchronization, active high
c. IVSM-(L) : Increment vertical state machine, active low
d. VIDEO-(L) : Displayed video refresh cycle, active low
e. LCC-(L) : Load column counter, active low
f. REFR-(L) : Refresh cycle, active low
g. CPU-(L) : Processor update cycle, active low

The timing diagram of these signals shown in Fig. 5.2(a) and (b), has been drawn assuming that the processor request for frame buffer memory access is always true i.e. CRQ-(H) is '1' and zoom factor bits ZFØ and ZF1 are both '0'. However, in an actual circuit operation,when no processor request is pending (i.e. CRQ-(H) is '0') the free cycles are allotted for the purpose of memory refresh. The zoom control bits decide the rate at which the frame buffer memory is required to be accessed for the purpose of refreshing the displayed image. The complete operation of the circuit for zoom is explained later in this chapter.

The VSM is incremented by the IVSM-(L) occurring during the HS79 i.e. last state of every line time. The signal LCC-(L) is generated in the last video cycle of a horizontal line time to load the starting value into the X-address counter as mentioned earlier in Sec. 2.2.3 to reset the column address. As can be seen from Fig. 5.2(b) no LCC-(L) pulse is generated during the time VBLNK(L)=0and also no video cycles are generated. During vertical active time,Video-(L) cycles are generated in every alternate cycle of the horizontal active time. The microprogram code for the HSM is provided in Appendix B.

5.2.3 Vertical State Machine (VSM)

The VSM is similar in construction to the HSM. Here a 9-bit address is provided by the microprogram address counters U4, U5, U8 to sequence the machine through its 315 states VSØ thru VS314. The microprogram address counter is incremented by the positive edge of INC-(H) in HS79 during which IVSM-(L) is active. The microprogram memory U9 (EPROM 2716) data is strobed into the eight bit latch U13 by the positive edge of LATCH-(L) during HS79. Thus the VSM output signals are stable for one line time. The format of the control signals generated by the VSM is given below :

a. VBLNK-(L) : Vertical blanking, active low

b. VSYNC-(L) : Vertical synchronization, active low

c. LRC-(L) : Load video refresh row address counter, active low

d. PERM-(L) : Permits the video refresh row address counter to be incremented, active low.

e. LLBLNK-(L): (NOT USED) Last line of vertical blanking time, active low

f. LLA-(L) : Last line of the vertical active time, active low.

The timing diagram of the signals at the VSM output is given in Fig. 5.3. The Y-address of the video refresh is reloaded into the row address counter on completion of VS 314 by the LLA-(L). When no zoom is employed (i.e., ZFØ=0, and ZF1 = 0) the video refresh row address counter is permitted to be incremented in every vertical state of the vertical active time by PERM-(L). However, when zoom is employed, the zoom factor bits ZFØ and ZF1 decide the rate at which the Y-address counters are incremented. The timing diagram of Fig. 5.3 has been drawn for zoom factor zero. The microprogram code for VSM is given in Appendix B.

5.2.4 Memory Refresh Circuit

This circuit consists of two 4-bit synchronous counters U58 and U64 (74LS163) and a 8-bit latch US9 (8282). The two counters provide the 7-bit address for generating the 128 addresses for memory refresh. During every refresh cycle (i.e. REFR-(L)) the counters are incremented by the INC-(H)

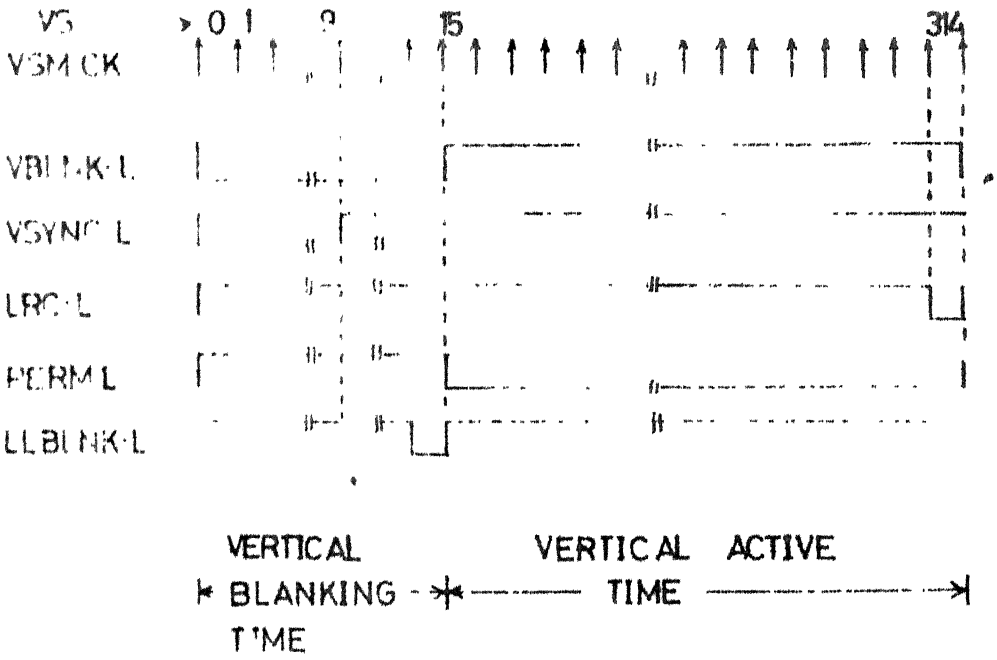


FIG 5·3 TIMING DIAGRAM OF VSM

and the address is strobed by LATCH-(H) . The refresh address is put on the DRAM address bus by controlling the output unable of U59 by REFR-(L) signal.

The HSM microprogram ensures that the requirement of generating 128 refresh cycles every 2 ms is complied with. During vertical blanking time atleast every alternate cycle is earmarked for memory refresh. During vertical active time atleast 29 memory refresh cycles are generated in every line time. The refresh address counters do not need to be loaded or cleared at any time due to the fact that the counter output will be continuously sequencing from 0 thru 127.

5.2.5 Video Refresh Circuit

The video refresh circuit consists of the X-address (i.e. column address) counters U63 and U68, and the Y- address (i.e. row address) counters U67, U65 and U66. The 5-bit column address sequences between 0 to 24 and is reset to the starting value during the last video cycle of every horizontal active time by load column counter signal LCC-(L). The column clock is generated by INC-(L) in every video cycle. The 9-bit row address sequences between 0 to 299. The row counter is reset to the starting value by the combination of load row counter signal LRC-(L) and LCC-(L). The combined 14-bit address (X and Y address) is used to access the particular 16-bit word from the memory for displayed video refresh. The row counter

is incremented by the row clock generated by INC-(L) in every HS 79 of horizontal active time. As will be explained later in this chapter, the Y-axis zoom is controlled by PERM-(L). The processor can write fresh starting address X' and Y' such that

$$0 \leq X' \leq 24 \quad \text{and} \quad 0 \leq Y' \leq 299 \ .$$

Thus panning is achieved by writing into latches U60 and U73. The zoom-Pan write logic ensures that the appropriate byte is written into U 60 and U 73. These two latches are I/O mapped with address bit AD7 low, (i.e. I/O address ØØH to 7FH). The 14-bit video refresh address is latched into latches U61 and U62 with LATCH-(H). This 14-bit address is multiplexed on to the 7-bit DRAM address bus by MUX-(H) and VIDEO-(L) signal (i.e. LAVIDEO-(L) and HAVIDEO-(L)).

5.2.6 Frame Buffer Memory

The frame buffer memory consists of U43 to U57 (i.e. 16 chips of DRAM 2118. The memory array is divided into Lower Bank and the Higher Bank. The Memory Bank Select Logic connects the processor data bus to the appropriate bank by decoding address bit ADØ. The $\overline{RD}$ signal of the processor controls the output enable of the DRAM buffers U70 and U69. The frame buffer memory is mapped at address space 8000H to FFFFH. Processor address bits AD14 to AD_1 are used to address the memory. For 400x300 display, address range 8000H to CAF1H is required to access each word.

During the video refresh cycle the two 8-bit parallel-in serial-out shift registers U31 and U38 are loaded by the S/LOAD-(L) pulse generated in every video cycle. The SR-CK generated from the X-axis Zoom Control logic is used to shift the data from the shift registers. During a processor read cycle the 16-bit data is latched into U70 and U69. However, only 8-bit data is routed to the processor data bus depending upon the signals generated by the memory bank select logic. During the processor write cycle the DRAM Write Logic generates the write signal WLBANK-(L) or WHBANK-(L) and the processor data bus is routed to the appropriate bank by the memory bank select logic. The direction of the bidirectional driver U71 and U72 (8286) is controlled by the processor generated $\overline{RD}$ signal. Thus the processor can access the frame buffer memory to read or write a 8-bit word.

5.2.7 Processor Interface

The processor's request for accessing the frame buffer is taken care of by the Processor Request Circuit and the Frame Buffer Select logic. The frame buffer select logic generates the signal BD-(L) depending upon the status of AD_{15} and IO/$\overline{M}$. The RDY line is pulled low by the falling edge of ALE when BD-(L) is active. The output $\overline{Q}$ of U29 is used to generate the processor cycle request signal CRQ-(H). The processor request CRQ-(H) is pulled low when a CPU cycle is in progress i.e. when the request has been honoured. The processor address is

multiplexed on to the 7-bit DRAM address bus by LACPU-(L) and HACPU-(L) signals generated in the Processor Address Buffer circuit. When the action required by the processor i.e. read/write is over at the end of CPU cycle the data is latched into memory buffers U69 and U70 by the LATCH-(H) pulse. The falling edge of LATCH-(L) is used to preset the F/F U29 and thus pulls the RDY line high.

The timing diagram shown in Fig. 5.4 exhibits the sequence of events leading to the generation and completion of a CPU cycle.

5.2.8 Video Generation Circuit

This circuit accepts the blanking, synchronisation, and video data signals to drive the CRT. HBLNK-(L), VBLNK-(L) are combined with the serial data of the shift registers to provide the composite video signal. The composite video, HSYNC-(L) and VSYNC-(L) are latched in a Quad D-type edge triggered F/F U15 (74175) by the pixel clock. The latched data is used to drive the CRT through a line driver U20 (74128). The composite video and HSYNC-(L) are connected by a shielded wire to the CRT. A jumper option enables the selection of NORMAL or REVERSE video data.

5.2.9 Zoom Control and Panning

The zoom control bits ZFØ and ZF1 allow zooming by effectively increasing the size of dots on the screen. This is

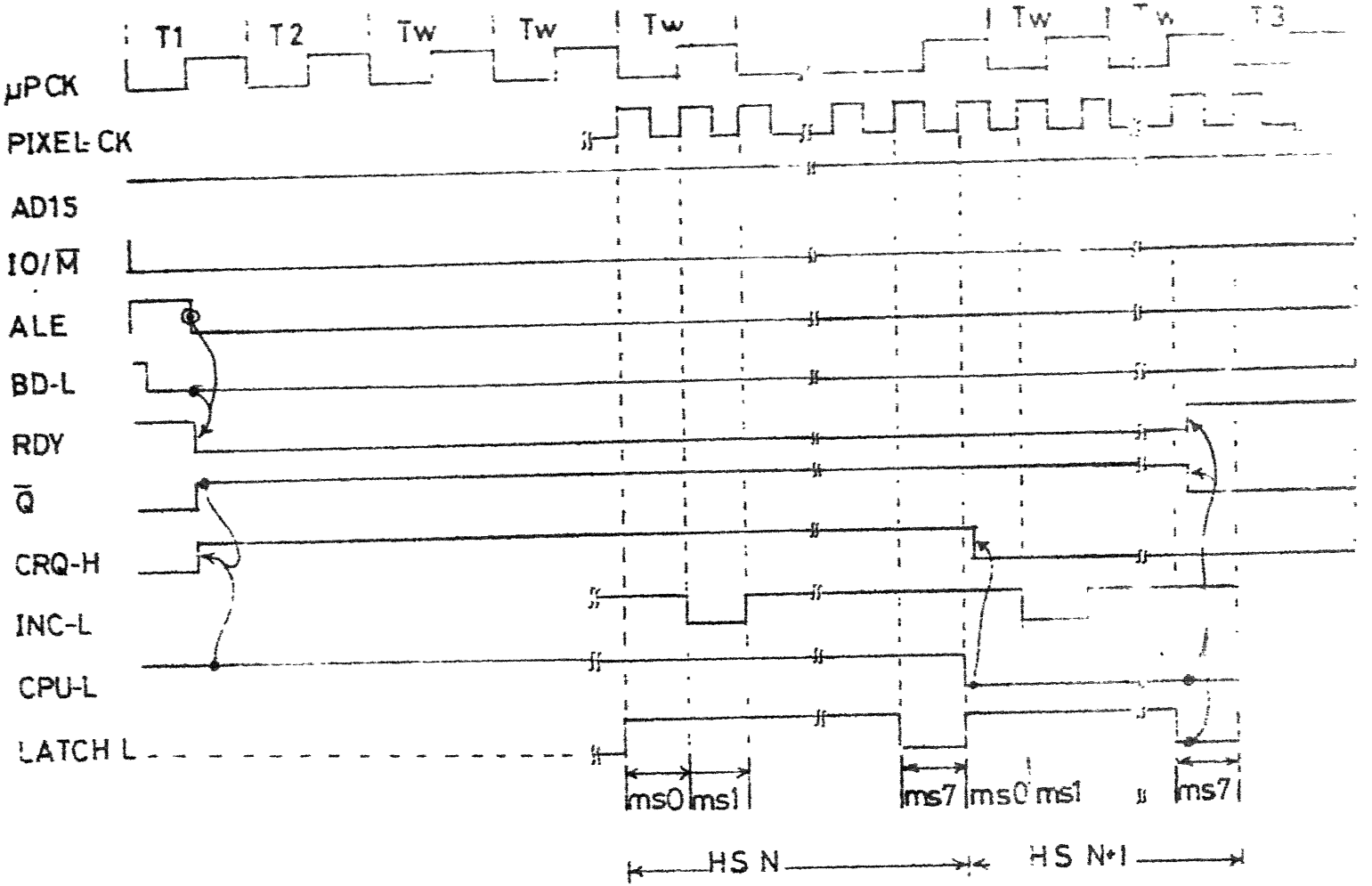
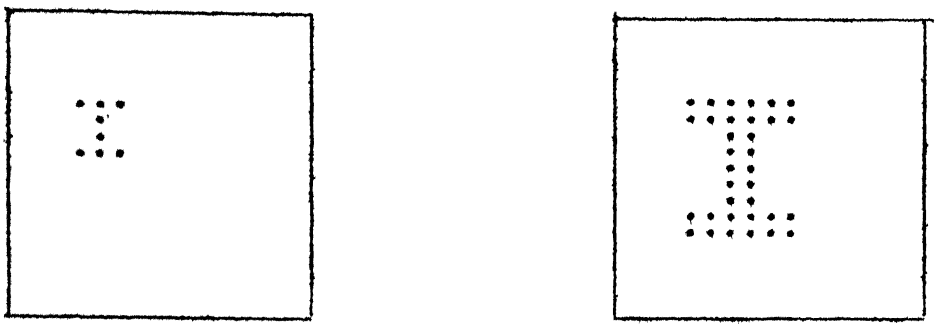


FIG 5-4 TIMING DIAGRAM FOR HONOURING PROCESSOR REQUESTS

accomplished vertically by repeating the same display line. The number of times it is repeated is determined by the display zoom factor parameter. Horizontally, zoom is accomplished by extending each display word cycle and displaying fewer words per line, according to the zoom factor. For example, consider Fig. 5.5(a) as showing an image displayed on the CRT. Fig. 5.5(b shows the same image magnified by a zoom factor of 2.



(a) (b) Zoom Factor-2

Fig. 5.5 Effect of Zooming on the displayed image

Magnification in the X-axis is achieved by slowing down the shift register clock SR-CK and the video refresh cycles. In the Y-axis, the row address counter clock ROW-CK is appropriately reduced.

In normal operation of the circuit,the fresh data is loaded into the shift register synchronously when S/R load signal is low during a video cycle. Timing diagram shown at Fig. 5.6 shows the sequence of operations. When zooming is not used each pixel is displayed for 100 ns and every 1.6 μs the shift register is loaded with fresh data. By appropriately

slowing down the pixel clock of 10 MHz, the duration for which each pixel value of the parallel-in, serial-out shift register is visible can now be prolonged. For example if the pixel clock is divided by 2 then each bit of the serial-out data of shift register is now stable for 200 ns. Thus magnification is achieved in the X-axis. It may be noted that during zooming the contents of the frame buffer are unaltered. Since, now the 16-bits of the shift register will take a longer time to be shifted out the rate at which video cycles are being generated should also be reduced by the same factor by which the clock has been reduced.

Since the loading operation of a the 74LS166 is synchronous in nature it must somehow be ensured that, irrespective of the zoom factor, a positive transition of the SR-CK must occur while SR load is low. This has been achieved by loading the pixel clock divide counter U28 with 1010 and then inverting the outputs before they are fed to the multiplexer. The loading of pixel clock divide counter is done in the video cycle occuring during every horizontal blanking time i.e. the first video cycle in HS29, by the INC-(L) pulse.

The timing diagram in Fig. 5.6 shows the sequence of operations.

The zoom control bits ZFØ and ZF1 are also routed to the HSM and VSM. Based on the status of ZFØ and ZF1 the HSM generates video refresh cycles every 1.6 μs, 3.2 μs, 6.4 μs, or 12.8 μs for zoom factor of 0,2,4,8 respectively. The VSM controls the increment to the row address counter by the PERM-(L) signal. During zoom Ø and the row address counter is incremented in every vertical state of vertical active time (VS 15 to VS 314). For zoom factor 2 the row address is incremented in every alternate state of the VSM, i.e. in VS16, VS18 etc. The shift register of load signal S/Load-(L) is obtained by delaying the SR Load-(L) pulse. This delay has been determined experimentally.

Panning is accomplished by changing the starting address of the display window. In this way panning is possible in any direction, vertically on a line by line basis and horizontally on a word by word basis. It is the responsibility of the processor to write the starting address and the zoom factor in the latches U60 and U73.

Fig. 5.7 shows the number of pixels and display lines which can be mapped on the CRT display for various zoom factors. This figure is drawn for starting address X = Ø, and Y = Ø. However, the display can be zoomed and the partitioned screen areas can be independently panned.

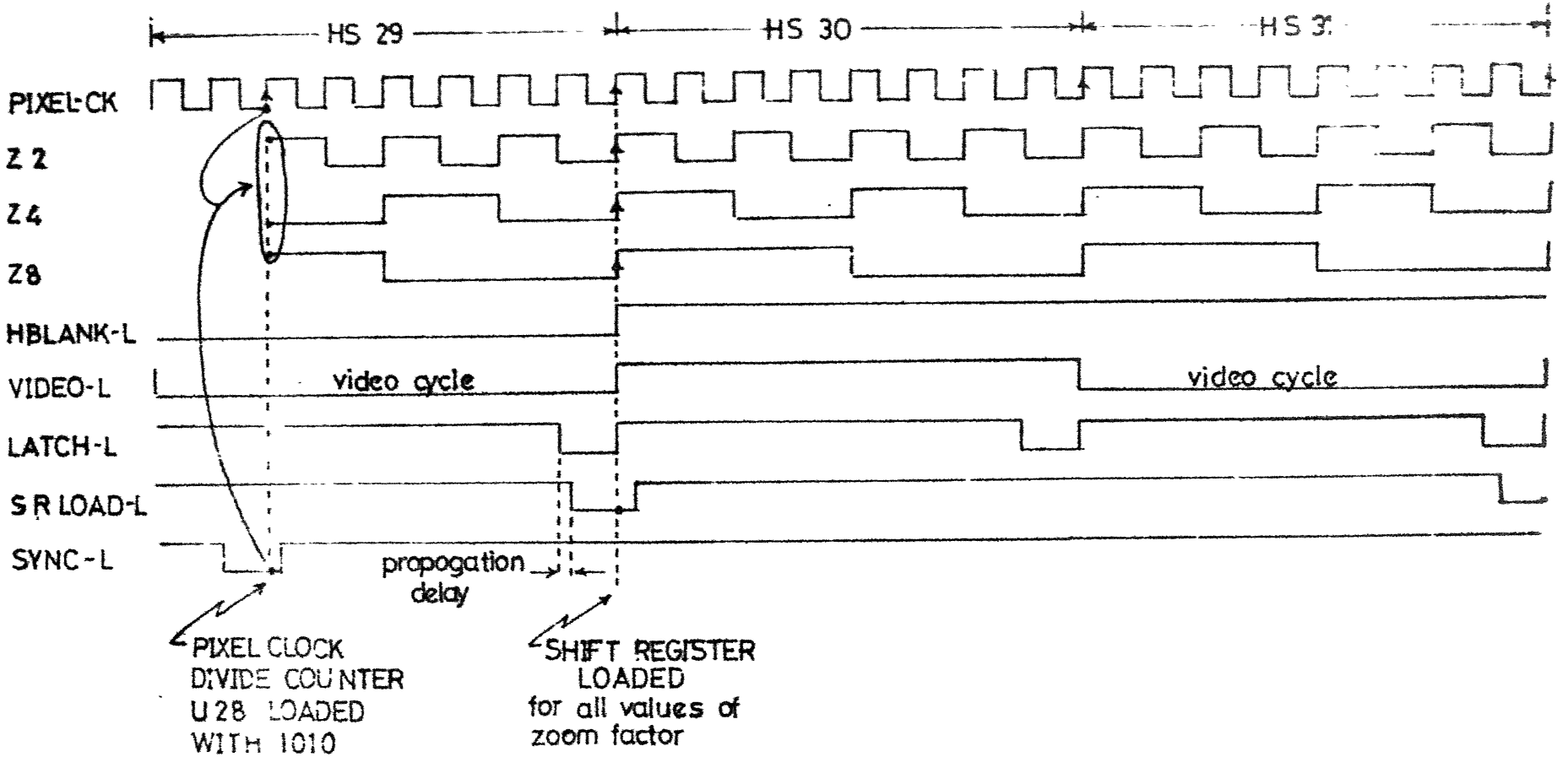


FIG 5-6 TIMING DIAGRAM OF SHIFT REGISTER LOADING AND SHIFT REGISTER CLOCK SYNCRONIZATION

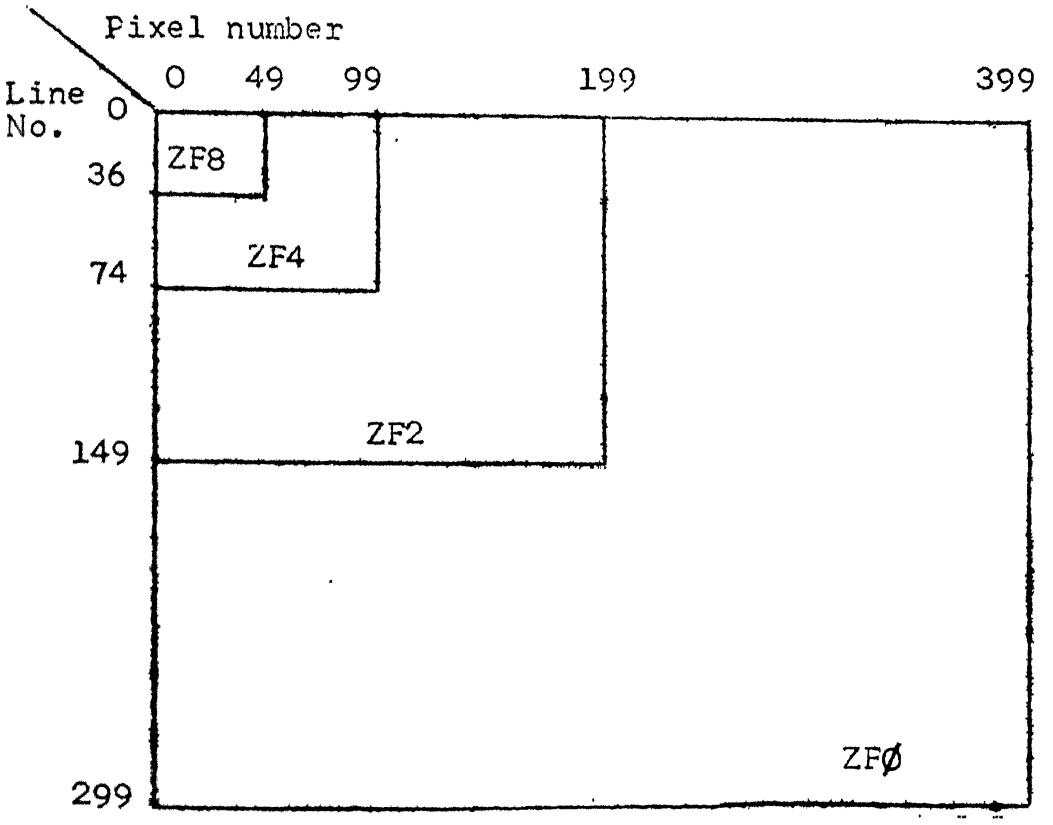


Fig. 5.7 Address map of frame buffer

dividing the pixel clock, the time for which each pixel is displayed can be altered. For zoom factor 2, the 5 MHz clock is used to clock the shift register thus each pixel is now displayed for 200 ns. In this case, the 16-bit data of the shift register will require 3.2 μs (16x200) to be shifted out. Therefore, now the shift register must be loaded every 3.2 μsec. In a similar fashion for the zoom factor of 4 and 8 the shift register is required to be loaded every 6.4 and 12.8 μsec respectively. However, it may be noticed that the first video cycle must always occur during the horizontal state HS29, at the end of which the horizontal blanking is removed. Thus the first bit of the shift register data is ready to be displayed the moment blanking is removed.

The shift register clock should be synchronized such that a positive transition of the clock must occur while S/R load is low, in order to synchronously load the shift register, whatever be the value of the zoom factor. The X-axis Zoom control logic divides the pixel clock and ensures that a positive transition of the clock occurs at the end of macrostate mS_7 in HS29. This synchronization is achieved by synchronously loading the clock divide counter U28 with 1010 by the INC-(L) pulse during the first video cycle (i.e. in HS 29). The inverted outputs of the U28 are fed to the multiplexer U27 (74153). The two bit zoom factor selection bits ZFØ and ZF1 are applied to the mux to select the appropriate clock for the shift register.

5.2.10 Microcode Explanation

The microcode for the HSM microprogram memory is provided, in Appendix B of this report. In this section we take a sample of the microcode for the purpose of explaining various signals being generated by the HSM and VSM.

The address pins of the HSM 2716 U7 are connected to the following signals.

a. A6-AØ - 7-bit microprogram address counter output to count 0_D to 79_D.

b. A7 - Vertical blanking signal from VSM

c. A8 - Processor cycle request CRQ (H)

d. A9 - ZFØ } Zoom Factor Select

e. A10 - ZF1 }

The selection of zoom by the signals ZFØ and ZF1 is as follows.

ZFØ	ZF1	Zoom Factor
0	0	No zoom
0	1	2
1	0	4
1	1	8

The combination of all the 11 address inputs A_{10}-AØ decide the address of memory location to be accessed.

Fig. 5.8 shows a part of the HSM microcode. Column 1 gives the microprogram address counter output. Column 2 indicates the time when the processor request is active, i.e., CRQ-(H). Fig. 5.8(a) shows the code when VBLNK-(L) input is '0' i.e. the HSM is generating signals during the vertical blanking time. It can be seen that every alternate cycle is reserved for memory refresh and processor access. The IVSM-(L) is programmed at address counter count 78. Signals for HSØ are programmed at count 79.

The following can be observed from Fig. 5.8 for all input conditions :

a. The first video cycle occurs in HS29 (therefore, it is programmed at EPROM memory location having counter address 28), during vertical active time.
b. IVSM-(L) is generated in HS79.
c. The horizontal blanking is removed with the HS 30.
d. HSØ always has identical signals (HEX code 5E) irrespective of input conditions.
e. The LCC-(L) is generated in HS79 when VBLNK-(L) = 1.

It can also be observed that with zoom factor 0 the video cycles are generated in HS28,30,32,34,36, etc and that with zoom factor 2 the video cycles are generated only at

		VBNKK-L=0								VBLNK-L = 1																													
ZF		0,2,4,8								0								2								4								8					
COU-NT	CRQ	NC	CPU	REFR	VIDEO	HSYNC	LLA	IVSM	HBLNK	NC	CPU	REFR	VIDEO	HSYNC	LLA	IVSM	HBLNK	NC	CPU	REFR	VIDEO	HSYNC	LLA	IVSM	HBLNK	NC	CPU	REFR	VIDEO	HSYNC	LLA	IVSM	HBLNK	NC	CPU	REFR	VIDEO	HSYNC	LLA	IVSM	HBLN
27	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0
28	1	0	0	1	1	1	1	1	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0
29	1	0	1	0	1	0	1	1	0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	1	0	1	0	1	1	1
30	1	0	0	1	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	0	1	1	0	1	1	1
31	1	0	1	0	1	0	1	1	0	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1
32	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1
33	0	0	1	0	1	0	1	1	0	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1
34	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1
35	1	0	1	0	1	0	1	1	0	0	0	1	1	0	1	1	1	0	1	0	1	0	1	1	1	0	0	1	1	0	1	1	1	0	1	0	1	0	1	1	1
36	1	0	0	1	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	0	1	1	1	0	0	1	1	0	1	1	1
37	1	0	1	1	0	1	1	1	0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	1	0	1	0	1	1	1
38	1	0	0	1	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	0	1	1	0	1	1	1
⋮																																										
74	0	0	1	1	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1
75	0	0	1	1	1	0	1	1	0	0	0	1	1	0	1	1	1	0	1	0	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1	1	1	1	1	1
76	0	0	1	1	1	0	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0	0	1	1
77	1	0	1	1	1	0	1	1	0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	1	0	1	0	1	1	1	0	0	1	1	0	1	1	1
78	1	0	0	1	1	0	1	1	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	1	0	1	0	0	0	1
79	1	0	1	1	1	1	1	0	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0
		a								b								c								d								e					

FIG 5-8 SECTION OF HSM MICROCODE

HS28, 32, 36, etc. Similarly the rate at which video cycles are generated is reduced for zoom factor 4 and 8. During vertical blanking time (i.e. Fig. 5.8(a)) no video cycles are generated. However, when VBLNK-(L) is active the states HSØ to HS 28 generate only memory refresh cycles. Between HS 29 and HS 79 the time of video cycle generation is fixed. The unused slots can be taken for generating memory refresh cycles if no processor request is pending i.e. CRQ-(H) is low, else the cycles are used forprocessor access.

Fig. 5.9 shows a section of VSM microprogram memory microcode. A 9-bit address is provided on pins A8-AØ of the EPROM 2716. The zoom factor control bits ZFØ and ZF1 are connected to A9 and A10 respectively. The microprogram address counter sequences from 0_D to 314_D in order to generate vertical states VSØ thru VS314. Fig. 5.9 shows the microcode for a given microprogram address counter output and various combinations of input conditions. All signals generated by the VSM are active low. As in HSM, the outputs required at vertical state N are programmed at program address counter output N-1. Thus signals required at VSØ are shown at counter count 314. When zoom factor is zero, the PERM-(L) is always active during the vertical active time. For zoom factor 2, the PERM-(L) is active only at VS16,VS18 etc., thus the clock to the row address counter is appropriately reduced. Similarly, for zoom factor 4 the PERM-(L) is active at VS 18, VS 22 etc. LLA-(L)

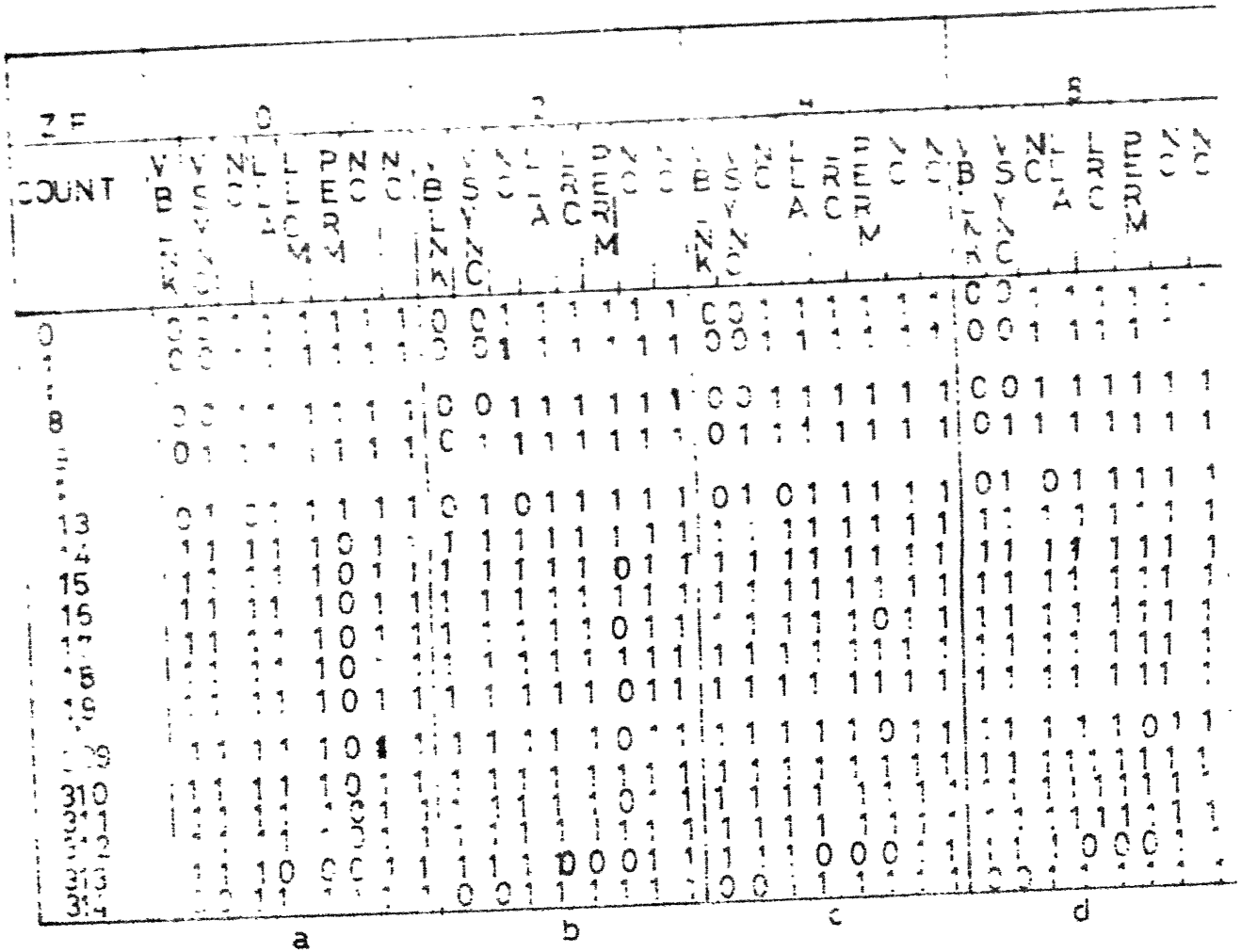


FIG 5-3 SECTION OF VSM MICRO CODE

is programmed at VS 314 to indicate the end of a scan time. Except for the PERM-(L) signal all signals are identical in Fig. 5.9 for the various zoom factors.

c. The 8-bit latches 8282 on the frame buffer card should be replaced by D-type positive edge triggered latches (74LS374) in order to ensure that data is available at the output on a known edge of strobe signal.

d. The critical signals should be properly guarded against noise by running them between ground lines. These signals include AØ, A15, IO/$\overline{M}$, $\overline{Rd}$, $\overline{WR}$, Pixel clock, Shift register data i.e. Video data, column clock, Row clock, Refresh clock, Column load and Row load.

e. In the present set up, 75 IC's have been used. The chipcount can be greatly reduced by replacing the combinatorial logic elements (i.e. NAND, NOR, OR gates etc.) by bipolar PROM's or PLA's to generate the control signals.

f. The Video data and Zoom factor data should be buffered before they are sent to the control card.

g. Wires between edge connectors should be avoided by using a motherboard arrangement.

h. The signals required to drive the CRT should be taken from a separate edge connector on the control card.

i. Panning could not be implemented due to noise pick-up on the $\overline{WR}$ line. For the present set up, X-Y starting address has been shorted to ground and the zoom factor control is achieved manually by setting the values of ZFØ and ZF1 to VCC or Gnd as needed.

h. High frequency signals like Pixel clock and Video data should be connected via shielded wire or twisted pair to avoid noise pick up.

6.2 SUGGESTIONS FOR FURTHER DEVELOPMENTS

The hardware implemented in this project amply demonstrates the working principles of a raster graphics system. A raster graphics system is a collection of hardware and software designed to make it easier to use graphics input and output in computer programs. The present hardware set-up can be fully used to develop a graphics package. Software development for this purpose is considerably simplified by the Microprocessor Development System.

The hardware can be expanded to implement a Multiple-Plane frame buffer to support greyscale or colour graphics. For this purpose the present set-up would have to be modified to provide additional memory along with the processor interface to access each plane of the frame buffer independently. Fig. 6.1 shows such a set-up. The display controller can now parallely read data from each memory plane and load the shift registers. The 4-bit per pixel data is used to access a look uptable, which can be independently programmed by the processor, to select the colour/intensity of each pixel. With a 1kx8 look up table 16 (out of 256 possible) colours/intensity values can be selected.

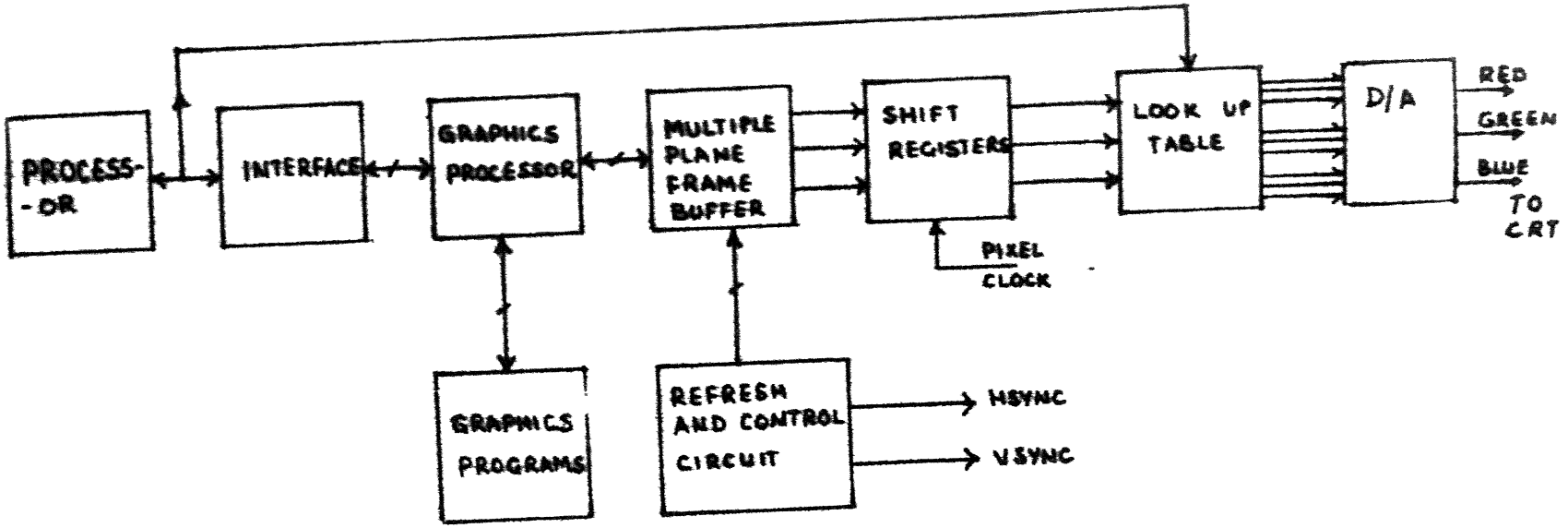


FIG 6·1 PROPOSED HARDWARE FOR DEVELOPMENT

Intel has recently announced a versatile Graphics display controller (GDC) Chip (82720) which considerably simplifies the raster graphics display hardware design. The GDC is an intelligent microprocessor peripheral designed to drive high performance raster scan computer graphic displays. When positioned between the frame buffer memory and the processor, the GDC performs the tasks needed to generate raster display and manage the frame buffer. Processor over head is minimised by the GDC's sophisticated instruction set, graphic figure drawing and DMA transfer capabilities. The GDC is capable of drawing characters, points, lines, arcs and rectangles and supports monochrome, greyscale or colour displays. The frame buffer memory supported by GDC can be configured in any formats and size upto 256K 16-bit word. The GDC has a 8-bit data bus for processor interface and a 16-bit internal structure. The GDC can support dot shift (pixel clock) rate of 80 MHz.

6.3 ELEMENTARY GRAPHICS SOFTWARE

In order to demonstrate the power of interactive raster graphics terminal, elementary software has been developed in PL/M80 on the Microprocessor Development System. Appendix D contains a listing for a PL/M-80 program which implements a set of primitive graphics routines. These routines are useful for forming many kinds of bit mapped images. The X,Y coordinates (to specify points, end points of lines and centre of circle) are assumed to be in the appropriate range for the given zoom factor. The range of X is 0 to 399 and Y from 0 to 299 with no zoom.

The graphics software can plot points, lines, circles and rectangles. Breshenham's algorithm for implementing the digital differential analyzer (DDA) has been implemented for line drawing. Circle plotting is implemented by drawing 45° arc from the X axis and Y axis in the first quadrant and copying the pixel values in the other quadrants. The turtle mode provides the simplest of all graphic input devices. By typing 't' to the main command level, the turtle level is entered. In the turtle section of the program, a small rectangle is displayed which signifies the present location of the turtle. The turtle can be moved up, down, left, right by typing 'U', 'D', 'L', 'R' respectively. Additional single character commands allow the line, rectangle and circle drawing. The user has the option of dragging the drawn object in any of the four direction. The software also permits drawing rubber band lines whereby the line can be moved in drawn in any direction with one end fixed.

REFERENCES

1. William M. Newman, Robert F. Sproull : Principles of Interactive Computer Graphics, 1981.

2. J.D. Foley, A Van Dam : Fundamentals of Interactive Computer Graphics, 1982.

3. Zvi Kohavi, Switching and Finite Automata Theory.

4. Microprogramming Handbook : Advanced Micro Devices, 1977.

5. Graphics Display Controller (82720) data sheet : Intel 1983.

6. AP 131 - Intel 2164A 64K Dynamic RAM device description.

7. AP 75 - Application of Intel 2118 16K Dynamic RAM

8. AP 118 - Raster Graphics Techniques for Multibus Users.

9. AP 133 - Designing memory systems for microprocessors using Intel 2164-A and 2118 Dynamic RAMS

10. A High Resolution Raster Scan display : HP Journal, June, 1975.

APPENDIX - A

CRT SPECIFICATIONS

The detailed specifications of the CRT used in this project are given below.

A1 Physical Characteristics

Size	12" Diagonal
Deflection angle	90^{o}
Glass area	74 Sq.in
Implosion protection	Tension band with mounting lugs
Phosphor	P31
Anode voltage	Approx 11.5 kV
Display size	8"(M) x 5.75" (V) Nominal
Face	Direct etched

A2 Electrical Characteristics

a. Video — TTL positive going pulse ($4.0V_{p-p} \pm 1.5V$) Input impedance: More than 3.3 K Ohms shunted by 60 pf. Rise time and fall time 35 ns or less. Video bandwidth : 18 MHz

b. Horizontal — TTL positive going pulse ($4.0V_{p-p} \pm 1.5V$) Input impedance: More than 470 Ohms shunted by 40 pf. Pulse width: 22 to 30 µsec at 15.75 kHz. Scan frequency : 15.75 kHz $\pm$ 0.5 kHz.

d. Vertical linearity — Same as for horizontal linearity

A4. Controls

a. Brightness	External
b. Vertical hold	Internal
Vertical height	"
Vertical linearity	"
Horizontal width	"
Horizontal linearity	"
Focus static	"

Environment

Operating temperature + 5°C thru + 55°C

A5. Pin Connections

1. GND : HD return, power return
2. External Brightness
3. "
4. "
5. ARC GND
6. Horizontal input
7. +15V DC
8. Video input
9. Vertical input
10. GND: VD return, video return

c. Vertical : TTL negative going pulse (4.0 $V_{p-p}\pm$ 1.5V). Input impedance: More than 3.3 K Ohms shunted by 40 pf.
Pulse width : 5 to 1400 µsec.
(It is recommended that the pulse duration be less than the vertical blanking interval).
Scan frequency : 47 to 63 Hz
Retrace time : 450 µsec

d. Power 15V DC $\pm$ 0.2 V/0.8A or less, nominal.

A3. Picture Quality

a. Resolution 900 TV lines at center, 800 TV lines at corners at 40 foot lambart.

b. Geometric Distortion The perimeter of a full field of video display shall not deviate over its entirety from the specified rectangular display dimension by more than 1.5% of the video raster vertical dimension

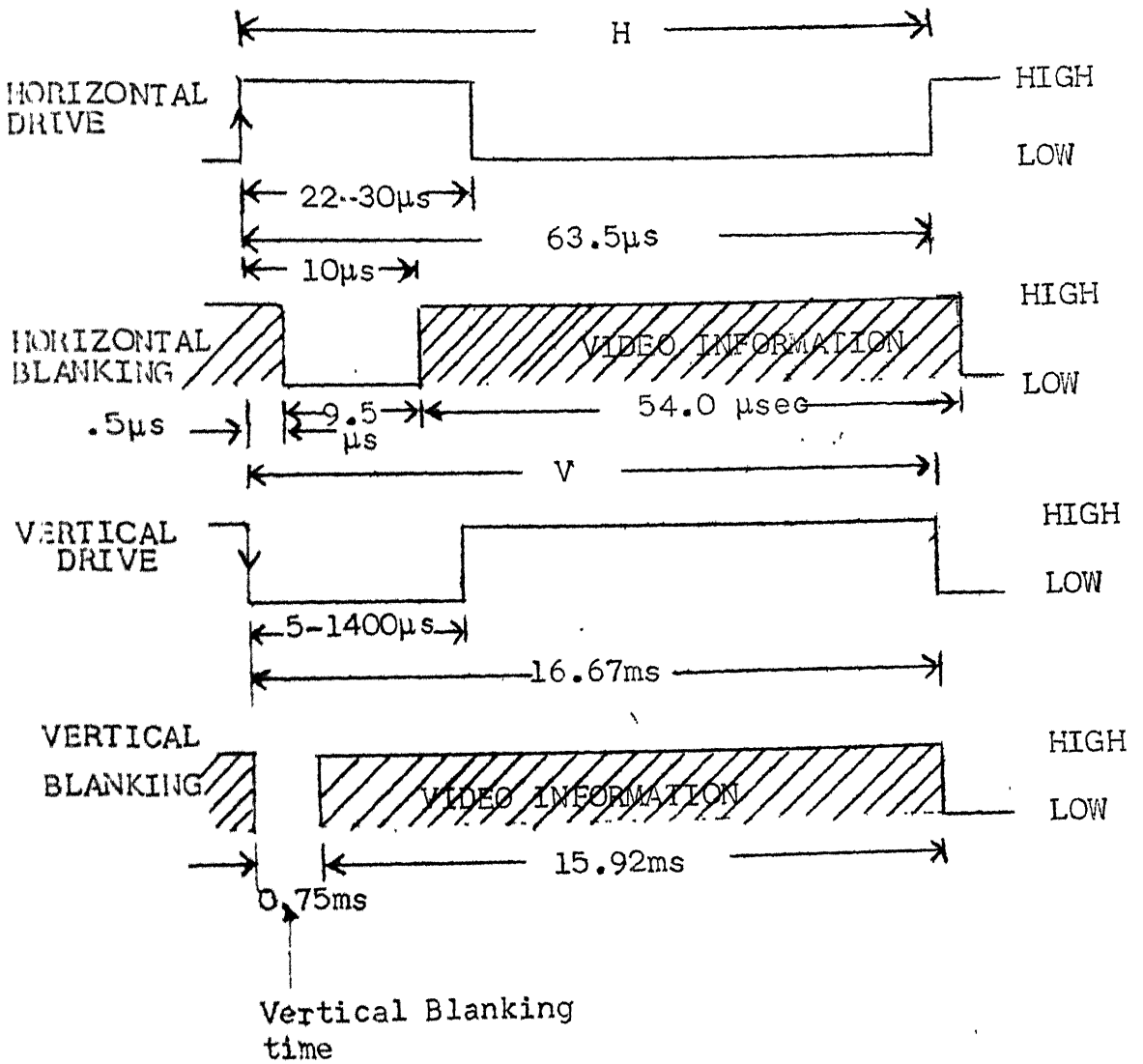
c. Horizontal linearity Less than 10% for adjacent characters. Less than 20% for any two entire display field.

A6 Timing Chart

At horizontal direct drive

H = 15750 Hz, V = 60 Hz

Vertical Retrace time = 450 µsec



APPENDIX B

: MICROCODE FOR

1) MACRO STATE MACHINE
2) HORIZONTAL STATE MACHINE
3) VERTICAL STATE MACHINE

MICRO CODE FOR MACRO STATE MACHINE

LOCATION (HEX)	COTENTS
10	CF
11	9F
12	CE
13	C4
14	C4
15	CO
16	CO
17	69

HORIZONTAL STATE MACHINE MICRO CODE

```
ADDRESS   :     CONTENTS
:10000000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
:10001000 5E5E5E5E5E5E5E5E5E5E5E5E5E565656
:10002000 5656565656565656565656565656565656
:10003000 5656565656565656565656565656565656
:10004000 565656565656565656565656565656545E
:10005000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10006000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10007000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10008000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
:10009000 5E5E5E5E5E5E5E5E5E5E5E5E6E576757
:1000A000 6757675767576757675767576757576757
:1000B000 6757675767576757675767576757576757
:1000C000 675767576757675767576757576357555E
:1000D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1000E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1000F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10010000 3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E
:10011000 3E5E3E5E3E5E3E5E3E5E3E5E3E563656
:10012000 3656365636563656365636563656563656
:10013000 3656365636563656365636563656563656
:10014000 365636563656365636563656365636345E
:10015000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10016000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10017000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10018000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
:10019000 5E5E5E5E5E5E5E5E5E5E5E5E6E376737
:1001A000 6737673767376737673767376737376737
:1001B000 6737673767376737673767376737376737
:1001C000 673767376737673767376737637355E
:1001D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1001E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1001F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10020000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
:10021000 5E5E5E5E5E5E5E5E5E5E5E5E565656
:10022000 5656565656565656565656565656565656
:10023000 5656565656565656565656565656565656
:10024000 565656565656565656565656565656545E
:10025000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10026000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10027000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10028000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
:10029000 5E5E5E5E5E5E5E5E5E5E5E5E6E576757
:1002A000 6757675767576757675767576757576757
:1002B000 6757675767576757675767576757576757
:1002C000 675767576757675767576757576357555E
:1002D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1002E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1002F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10030000 3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E
```

```
: 10031000 3E5E3E5E3E5E3E5E3E5E3E5E3E563656
: 10032000 365636563656365636563656365636563656
: 10033000 365636563656365636563656365636563656
: 10034000 36563656365636563656365636563656345E
: 10035000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10036000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10037000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10038000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
: 10039000 5E5E5E5E5E5E5E5E5E5E5E5E5E6E376737
: 1003A000 673767376737673767376737673767376737
: 1003B000 673767376737673767376737673767376737
: 1003C000 67376737673767376737673763373556E
: 1003D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 1003E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 1003F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10040000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
: 10041000 5E5E5E5E5E5E5E5E5E5E5E5E5E565656
: 10042000 56565656565656565656565656565656
: 10043000 56565656565656565656565656565656
: 10044000 565656565656565656565656565656545E
: 10045000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10046000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10047000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10048000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
: 10049000 5E5E5E5E5E5E5E5E5E5E5E5E6E575757
: 1004A000 57575757675757575757575767575757
: 1004B000 57575757675757575757575767575757
: 1004C000 5757575767575757575757576357575E
: 1004D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 1004E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 1004F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10050000 3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E
: 10051000 3E5E3E5E3E5E3E5E3E5E3E5E3E563656
: 10052000 365636563656365636563656365636563656
: 10053000 365636563656365636563656365636563656
: 10054000 36563656365636563656365636563656345E
: 10055000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10056000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10057000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10058000 3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E
: 10059000 3E5E3E5E3E5E3E5E3E5E3E5E6E375737
: 1005A000 57375737673757375737573767573757
: 1005B000 37573757675737573757375767375737
: 1005C000 5737573767375737573757376357355E
: 1005D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 1005E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 1005F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
: 10060000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
: 10061000 5E5E5E5E5E5E5E5E5E5E5E5E5E565656
: 10062000 56565656565656565656565656565656
```

```
:10063000 56565656565656565656565656565656
:10064000 56565656565656565656565656565656
:10065000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10066000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10067000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10068000 5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E5E
:10069000 5E5E5E5E5E5E5E5E5E5E5E5E6E575757
:1006A000 57575757675757575757575767575757
:1006B000 57575757675757575757575767575757
:1006C000 57575757675757575757575763575757
:1006D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1006E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1006F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10070000 3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E
:10071000 3E5E3E5E3E5E3E5E3E5E3E5E3E563656
:10072000 36563656365636563656365636563656
:10073000 36563656365636563656365636563656
:10074000 36563656365636563656365636563656
:10075000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10076000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10077000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10078000 3E5E3E5E3E5E3E5E3E5E3E5E3E5E3E5E
:10079000 3E5E3E5E3E5E3E5E3E5E3E5E6E375737
:1007A000 57375737673757375737573767573757
:1007B000 37573757675737573757375767375737
:1007C000 57375737673757375737573763573556
:1007D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:00000001 FF
++++++++++++++++++++++++++++++++++++++++++++++++++++
```

VERTICAL STATE MACHINE MICROCODE

****

CONTENTS

```
:1002E000FBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFB
:1002F000FBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFB
:10030000FBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFB
:10031000FBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFB
:10032000FBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFBFB
:10033000FBFBFBFBFBFBFBFBFBFBE33FFFFFFFFFFFF
:10034000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10035000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10036000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10037000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10038000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10039000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1003A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1003B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1003C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1003D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1003E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1003F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:100400003F3F3F3F3F3F3F3F3F7F7F7F7F5FFFFF
:10041000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10042000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10043000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10044000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10045000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10046000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10047000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10048000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10049000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1004A000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1004B000FFFBFFFFFFFBFFFFFEFBFFFFFEFBFFFF
:1004C000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1004D000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1004E000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1004F000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10050000FFFBFFFFFFFBFFFFFEFBFFFFFFFBFFFF
:10051000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10052000FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10053000FFFBFFFFFFFBFFFFFFE33FFFFFFFFFFF
:10054000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10055000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10056000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10057000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10058000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10059000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1005A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1005B000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1005C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1005D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1005E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1005F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:100600003F3F3F3F3F3F3F3F3F7F7F7F7E5FFFFF
```

```
:10061000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10062000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10063000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10064000 FFFBFFFFFFFBFFFFFFFBFFFFFFFAFFFF
:10065000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10066000 FFFBFFFFFFFBFFFFFFFBFFFFFEFBFFFF
:10067000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10068000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10069000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1006A000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1006B000 FFFBFFFFFFFBFFFFFEFBFFFFFEFBFFFF
:1006C000 FFFBFFFFFFFBFFFFFFFBFFFEFFFBFFFF
:1006D000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:1006E000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFEFF
:1006F000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10070000 FFFBFFFFFFFBFFFFFFFAFFFFFFFBFFFF
:10071000 FFFBFFFFFFFBFFFFFFFBFFFFFFFBFFFF
:10072000 FFFBFFFFFFFBFFFFFEFBFFFFFFFBFFFF
:10073000 FFFBFFFFFFFBFFFFFFE33FFFFFFFFFFF
:10074000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10075000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10076000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10077000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10078000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:10079000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007A000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007B000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007C000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007D000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007E000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:1007F000 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
:00000001 FF
```

Edge Connector details

Graphics Control Card (44 pin)

Pin No.	TOP	BOTTOM
1	ZF1	GND
2	ZFØ	HSYNC
3	CRQ-(H)	VSYNC
4	GND	COM-VIDEO
5	SR Data	GND
6	GND	+5
7	LCC-(L)	VIDEO-(L)
8	COL-CK	INC-(L)
9	ROW-CK	HBLNK
10	LACPU-(L)	LATCH-(L)
11	MR-CK	
12	REFRESH-(L)	
13	LAVIDEO-(L)	
14	ROW-LOAD-(L)	
15	SRLOAD-(L)	
16	CPU-(L)	
17	HA CPU-(L)	
18	CAS-(L)	
19	RAS-(L)	
20	HA-VIDEO-(L)	
21	PIXEL-CK	
22	LATCH-(H)	

(a)

Work-Station (44 pin)

Pin No.	TOP	BOTTOM
1		
2	A_{15}	
3	A_{14}	D_7
4	A_{13}	D_6
5	A_{12}	D_5
6	A_{11}	D_4
7	A_{10}	D_3
8	A_9	D_2
9	A_8	D_1
10	A_7	D_0
11	A_6	IO/$\overline{M}$
12	A_5	$\overline{RD}$
13	A_4	$\overline{WR}$
14	A_3	
15	A_2	RDY
16	A_1	
17	$A_Ø$	
18		ALE
19		
20		
21		
22	GND	GND

(b)

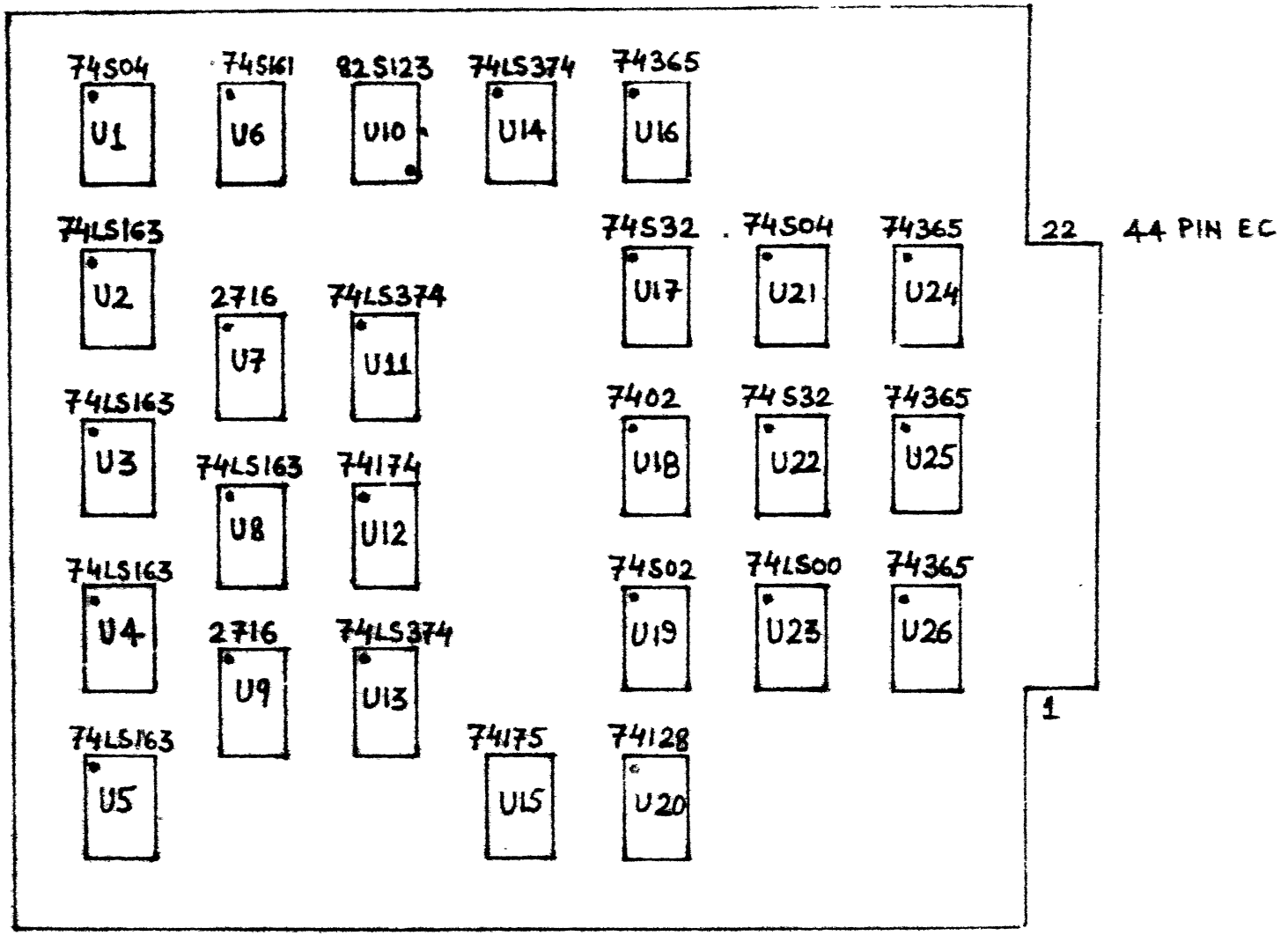
Frame Buffer Card (32 pin)

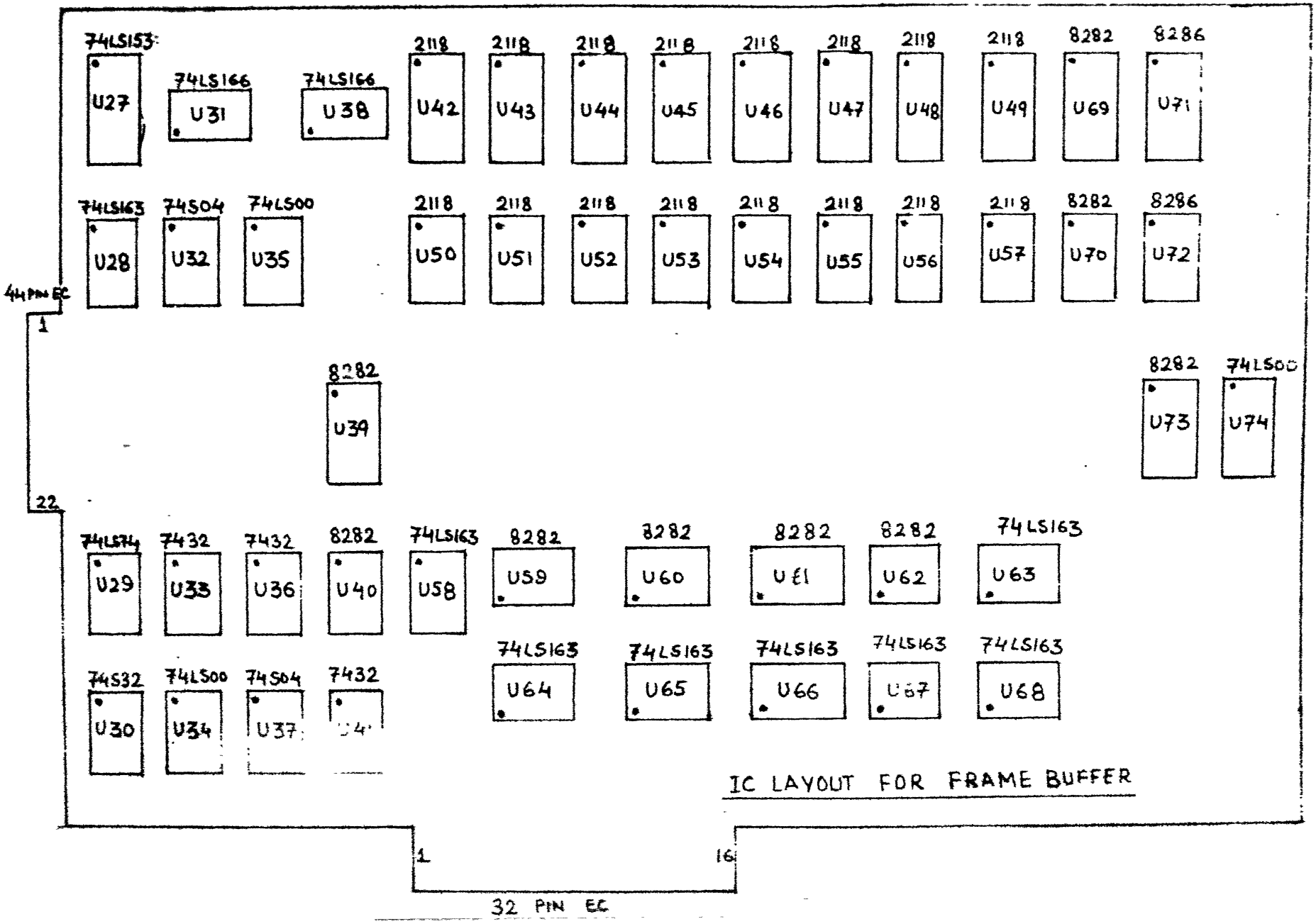
Pin No.	TOP	BOTTOM
1.	LA CPU-(L)	HACPU-(L)
2	SR DATA	CPU-(L)
3	VIDEO-(L)	INC-(L)
4	CRQ-(H)	HBLNK
5	WE-(L)	LATCH-(L)
6	CAS-(L)	RAS-(L)
7	MRCK	PIXEL CK
8	REFR-(L)	SRLOAD-(L)
9	ZF_1	LATCH-(H)
10	$ZF_\emptyset$	ROWCK
11	ROWLOAD-(L)	HAVIDEO-(L)
12	LATCH-(H)	LAVIDEO-(L)
13	COL CK	COLLOAD-(L)
14	+5V	GND
15		
16		

Frame Buffer Card (44 pin)

Pin No.	TOP	BOTTOM
1	D_7	D_6
2	D_5	D_4
3	D_3	D_2
4	D_1	$D_\emptyset$
5	A_{13}	A_{14}
6	A_{11}	A_{12}
7	A_9	A_{10}
8	A_7	A_8
9	A_5	A_6
10	A_3	A_4
11	A_1	A_2
12		
13	GND	+5V
14		
15		
16		
17		RDY
18		ALE
19		$\overline{RD}$
20		$\overline{WR}$
21		IO/$\overline{M}$
22	A_{15}	$A_\emptyset$

IC LAYOUT FOR GRAPHICS CONTROL CARD





IC LAYOUT FOR FRAME BUFFER

```
ISIS-II PL/M-80 V4.0 COMPILATION OF MODULE GRAPH
OBJECT MODULE PLACED IN :F2:GR2.OBJ
COMPILER INVOKED BY:  :F2:PLM80 :F2:GR2.1 WORKFILES(:F2:,:F2
                      -:) PAGEWIDTH(60) PAGELENGTH(45) DATE(4
                      -SEPT 84)


  1                 GRAPH :
                    DO;

  2     1           DECLARE (X,Y,SIZE)ADDRESS;
  3     1           DECLARE TRUE LITERALLY '0FFH';
  4     1           DECLARE FALSE LITERALLY '000H';
  5     1           DECLARE CR LITERALLY '00AH';
  6     1           DECLARE LF LITERALLY '00DH';
  7     1           DECLARE USTAT LITERALLY '0F9H';
  8     1           DECLARE UDATA LITERALLY '0F8H';
  9     1           DECLARE RXBF LITERALLY '002H';
 10     1           DECLARE TXBE LITERALLY '001H';
 11     1           DECLARE HEXCODE(*)BYTE INITIAL('0123456789ABCD
            -       EF');
 12     1           DECLARE CHAR BYTE ;
 13     1           DECLARE FB0(32768) BYTE AT(08000H);
 14     1           DECLARE DARK LITERALLY '000H';
 15     1           DECLARE BRIGHT LITERALLY '0FFH';
 16     1           DECLARE (SCREENWIDTH,SCREENHEIGHT) ADDRESS;
 17     1           DECLARE PLOTMODE BYTE;
 18     1           DECLARE (RUBBER,RB) BYTE;
 19     1           DECLARE (NX1,NY1,NX2,NY2) ADDRESS;
 20     1           DECLARE RADIUS ADDRESS;
 21     1           DECLARE (XORG,YORG) ADDRESS;

                      /* PROCEDURES GET AND PUT READ AND
                         WRITE RESPECTIVELY THE KEYBOAD
                         STAUS IN THE USART*/

 22     1           GET:PROCEDURE BYTE;
 23     2           DO WHILE (INPUT(USTAT)AND RXBF)<> RXBF;END;
 25     2           RETURN(INPUT(UDATA) AND 07FH);
 26     2           END GET ;
```

```
27   1       PUT:PROCEDURE(CHAR);
28   2       DECLARE CHAR BYTE;
29   2       DO WHILE (INPUT (USTAT) AND TXBE)<> TXBE;END;
31   2       OUTPUT (UDATA)= CHAR;
32   2       END PUT;

33   1       CRLF: PROCEDURE;
34   2       CALL PUT(CR);
35   2       CALL PUT(LF);
36   2       END CRLF;

              /* THIS PROCEDURE CONVERTS A BYTE
                 INTO ASCII AND SENDS TO TTY */

37   1       BYTEOUT:PROCEDURE(B);
38   2       DECLARE B BYTE;
39   2       CALL PUT (HEXCODE (SHR((B AND 0F0H),4)));
40   2       CALL PUT (HEXCODE (B AND 00FH));
41   2       END BYTEOUT;

42   1       WORDOUT:PROCEDURE(W);
43   2       DECLARE W ADDRESS;
44   2       CALL BYTEOUT(HIGH(W));
45   2       CALL BYTEOUT(LOW(W));
46   2       END WORDOUT;

              /* THIS PROCEDURE CLEARS THE FRAME BUFFER
         -    */

47   1       CLEARSCREEN: PROCEDURE (VAL);
48   2       DECLARE (VAL) BYTE;
49   2       DECLARE (J) ADDRESS;
50   2       DO J= 00H TO 04BF1H;
51   3       FB0 (J) = VAL;
52   3       END;
53   2       END;

              /* THIS PROCEDURE MODIFIES THE GIVEN PIXE
         -    L */


54   1       MODIFY:PROCEDURE(WORDADDR,PATTERN);
```

```
55   2        DECLARE (WORDADDR) ADDRESS;
56   2        DECLARE (PATTERN) BYTE;
57   2        DECLARE (MASK) BYTE;
58   2        DO CASE PLOTMODE;
59   3         FBO(WORDADDR) = FBO(WORDADDR) OR PATTERN;
60   3         FBO(WORDADDR) = FBO(WORDADDR) AND NOT PATTERN
               ;
61   3         FBO(WORDADDR) = FBO(WORDADDR) XOR PATTERN;
62   3        END;
63   2        END MODIFY;

             /* THIS PROCEDURE COMPUTES THE WORD ADDRESS
                AND THE BIT IN THAT WORD WHICH HAS TO BE
                MODIFIED. THE INPUTS ARE THE X-ADDRESS
                (0 < X<39 ) AND Y-ADDRESS (0< Y <299). *
               /
64   1        PLOTPT: PROCEDURE (AX, AY);
65   2        DECLARE (PMASK) BYTE;
66   2        DECLARE(PIXEL) BYTE;
67   2        DECLARE (YNEW) ADDRESS;
68   2        DECLARE (AX, AY) ADDRESS;
69   2        DECLARE (XADDR) ADDRESS;
70   2        DECLARE (YADDR) ADDRESS;
71   2        DECLARE (FBADDR) ADDRESS;
72   2        PIXEL =LOW (AX) AND 007H;
73   2        PMASK =001H;
74   2        IF PIXEL <> 0 THEN PMASK = ROL(PMASK, PIXEL);
76   2        XADDR =(SHR (AX, 3) AND 0003FH);
77   2        YNEW = 0012BH -AY;
78   2        YADDR =(SHL (YNEW, 6) AND 07FC0H);
79   2        FBADDR =(YADDR OR XADDR);
80   2        CALL MODIFY (FBADDR, PMASK);
81   2        END PLOTPT;


             /* THIS PROCEDURE ASCII OF HEX INPUT
                INTO THE HEX WORD. */

82   1        CODE: PROCEDURE (C) BYTE;
83   2        DECLARE C BYTE;
84   2        DECLARE D BYTE;
```

```
 85   2        IF ( C >='a') AND ( c <= 'f') THEN D=c-'a'+10
        -        ;
 87   2        ELSE IF (C>='A') AND (C<='F') THEN D=C-'A'+10;
 89   2        ELSE IF (C>='0') AND (C<='9') THEN D= C-'0';
 91   2        ELSE D=16;
 92   2        RETURN (D);
 93   2        END CODE;

                     /* THIS PROCEDURE READS IN A HEX CHAR */

 94   1        WORDIN: PROCEDURE ADDRESS;
 95   2        DECLARE W ADDRESS;
 96   2        DECLARE (C,CHAR) BYTE;
 97   2        W = 0; C=0;
 99   2        DO WHILE C<>16;
100   3        W=SHL(W,4)+C;
101   3        CHAR=GET;
102   3        CALL PUT(CHAR);C=CODE(CHAR);
104   3        END;
105   2        RETURN(W);
106   2        END WORDIN;

                    /* BREZENHAM'S ALGORITHM */

107   1        DDA: PROCEDURE(X1,Y1,X2,Y2);
108   2        DECLARE (X1,X2,Y1,Y2) ADDRESS;
109   2        DECLARE (I,X,Y,DELX,DELY,DELX2,DELY2,REM,DIREC
        -         TION) ADDRESS;
110   2        CALL CRLF;
111   2        IF Y1>Y2 THEN DO; Y=Y1;Y1=Y2;Y2=Y;
116   3        X=X1;X1=X2;X2=X;END;
120   2        DELY=Y2-Y1;
121   2        IF X1 > X2 THEN DO; DELX= X1-X2;DIRECTION=1;EN
        -         D;
126   2                        ELSE DO; DELX= X2-X1;DIRECTION=0;EN
        -         D;
130   2        IF (DELX=0) AND (DELY =0) THEN
131   2         DO;CALL PLOTPT(X1,Y1);RETURN;END;
135   2        DELX2 =SHL (DELX,1);
136   2        DELY2 =SHL (DELY,1);
137   2        Y=Y1;X=X1;
139   2        IF DELY <= DELX THEN REM=DELY2-DELX;
```

```
141   2                          ELSE REM=DELX2-DELY;
142   2       IF DELY=0 THEN
143   2       DO;
144   3       IF DIRECTION =0 THEN
145   3       DO X=X1 TO X2 ;
146   4       CALL PLOTPT(X,Y1);
147   4       END;
              ELSE
148   3       DO X=  X2 TO X1;
149   4       CALL PLOTPT (X,Y1);
150   4       END;
151   3       END;
152   2       ELSE IF DELX =0 THEN
153   2       DO Y =Y1 TO Y2;
154   3       CALL PLOTPT(X1,Y);
155   3       END;
156   2       ELSE IF DELY <= DELX THEN
157   2       DO I=0 TO DELX;
158   3       CALL PLOTPT(X,Y);
159   3       IF REM < 32768 THEN
160   3       DO;
161   4       Y = Y + 1;
162   4       REM = REM + DELY2 - DELX2;
163   4       END;
164   3       ELSE REM =REM +DELY2;
165   3       IF DIRECTION = 0 THEN X = X+1;ELSE X=X-1;
168   3       END;
169   2       ELSE IF DELY > DELX THEN
170   2       DO I=0 TO DELY;
171   3       CALL PLOTPT(X,Y);
172   3       IF REM < 32768 THEN
173   3       DO;
174   4          IF DIRECTION = 0 THEN X=X+1; ELSE X= X-1;
177   4         REM = REM +DELX2 -DELY2;
178   4       END;
179   3       ELSE REM = REM+DELX2;
180   3       Y =Y+1;
181   3       END;
182   2       END DDA;

            /* THIS PROCEDURE PLOTS A RECTANGLE
               GIVEN TWO DIOGNAL POINTS          */
```

```
183   1       LINERECT:PROCEDURE(X1,Y1,X2,Y2);
184   2       DECLARE(X1,Y1,X2,Y2)ADDRESS;
185   2       CALL DDA(X1,Y1,X2,Y1);
186   2       CALL DDA(X2,Y1,X2,Y2);
187   2       CALL DDA(X1,Y2,X2,Y2);
188   2       CALL DDA(X1,Y1,X1,Y2);
189   2       END LINERECT;

                    /* CIRCLE PLOTTING PROCEDURE*/

190   1       ARC$PLOT:PROCEDURE(Xn,Yn);
191   2       DECLARE (Xn,Yn,XT,YT) ADDRESS,
192   2       Xn=Xn + XORG;Yn=Yn+YORG;
194   2       XT=Xn;YT=Yn;
196   2       CALL PLOTPT(Xn,Yn);
197   2       Xn=SHL(XORG,1)-Xn;
198   2       CALL PLOTPT(Xn,Yn);
199   2       Xn=XT;
200   2       Yn=SHL(YORG,1)-Yn;
201   2       CALL PLOTPT(Xn,Yn);
202   2       Xn=SHL(XORG,1)-Xn;
203   2       CALL PLOTPT(Xn,Yn);
204   2       END ARC$PLOT;

205   1       ARCA:PROCEDURE(R);
206   2       DECLARE(Xn,Yn,R,YY,Si,XX) ADDRESS;
207   2       Xn=R;
208   2       XX=SHL(Xn,1);
209   2       YY=1;Yn=0;
211   2       Si=SHR(XX,1);
212   2       DO WHILE (YY<XX);
213   3        CALL ARC$PLOT(Xn,Yn);
214   3        IF Si>=XX THEN
215   3         DO;
216   4           Xn=Xn-1;
217   4           Si=Si-XX;
218   4         END;
219   3        Si=Si+YY;
220   3        Yn=Yn+1;
221   3        XX=SHL(Xn,1);
222   3        YY=((SHL(Yn,1)+1));
```

```
223   3        END;
224   2       END ARCA;


225   1       ARCB: PROCEDURE(R);
226   2       DECLARE(Xn, Yn, R, XX, YY, Si) ADDRESS;
227   2       Yn=R; YY=SHL(Yn, 1);
229   2       XX=1; Xn=0;
231   2       Si=SHR(YY, 1);
232   2       DO WHILE YY>XX;
233   3        CALL ARC$PLOT(Xn, Yn);
234   3        IF Si>=YY THEN
235   3        DO;
236   4         Yn=Yn-1;
237   4         Si=Si-YY;
238   4         END;
239   3        Si=Si+XX;
240   3        Xn=Xn+1;
241   3        YY=SHL(Yn, 1);
242   3        XX=(SHL(Xn, 1)+1);
243   3        END;
244   2       END ARCB;


245   1       CIRCLE : PROCEDURE(Xc, Yc, Rc);
246   2       DECLARE (Xc, Yc, Rc) ADDRESS;
247   2       XORG=Xc; YORG=YC;
249   2       CALL ARCA(Rc);
250   2       CALL ARCB(Rc);
251   2       END CIRCLE;
                /* TURTLE LEVEL PROGRAME */

252   1       TURTLE : PROCEDURE;
253   2       DECLARE CHAR BYTE;
254   2       DECLARE (X, Y, DELTA) ADDRESS;
255   2       DECLARE (PX1, PY1, PX2, PY2) ADDRESS;

256   2       TOGGLE : PROCEDURE(X, Y);
257   3        DECLARE (X, Y) ADDRESS;
258   3        CALL LINERECT (X-2, Y-2, X+2, Y+2);
259   3       END TOGGLE;
```

```
260   2       TOGGLELINE:PROCEDURE(X1,Y1,X2,Y2);
261   3       DECLARE (X1,Y1,X2,Y2) ADDRESS;
262   3       CALL DDA(PX1,PY1,PX2,PY2);
263   3       PX1=X1;PY1=Y1;PX2=X2;PY2=Y2;
267   3       CALL DDA(PX1,PY1,PX2,PY2);
268   3       END;

269   2       TOGGLE$RECT:PROCEDURE(X1,Y1,X2,Y2);
270   3       DECLARE (X1,Y1,X2,Y2) ADDRESS;
271   3       CALL LINERECT(PX1,PY1,PX2,PY2);
272   3       PX1=X1;PX2=X2;PY1=Y1;PY2=Y2;
276   3       CALL LINERECT(PX1,PY1,PX2,PY2);
277   3       END TOGGLE$RECT;

278   2       TOGGLE$CIRCLE:PROCEDURE(X1,Y1,R);
279   3       DECLARE (X1,Y1,R) ADDRESS;
280   3       CALL CIRCLE(PX2,PY2,R);
281   3       PX2=X1;PY2=Y1;
283   3       CALL CIRCLE(PX2,PY2,R);
284   3       END TOGGLE$CIRCLE;

285   2       RUBBER$LINE:PROCEDURE (X1,Y1,X2,Y2);
286   3       DECLARE (X1,Y1,X2,Y2) ADDRESS;
287   3       CALL DDA(PX1,PY1,PX2,PY2);
288   3       PX2=X2;PY2=Y2;
290   3       CALL DDA (PX1,PY1,PX2,PY2);
291   3       END;


292   2       CRAWLTO: PROCEDURE(XNEW,YNEW);
293   3        DECLARE(XNEW,YNEW) ADDRESS;
294   3        CALL TOGGLE(X,Y);
295   3        X=XNEW; Y=YNEW;
297   3        CALL TOGGLE(X,Y);
298   3       END CRAWLTO;

299   2       RB=00H;RUBBER=00;
301   2       PLOTMODE=2;
302   2       CHAR =0;DELTA=1;
304   2       X=200; Y=150;
306   2       CALL TOGGLE(X,Y);
```

```
307   2       DO WHILE (CHAR<>'E');
308   3        CHAR=GET;
309   3        CALL PUT(CHAR);
310   3        IF (CHAR='C') THEN DELTA=WORDIN;
312   3        IF (CHAR='U') THEN DO;
314   4                          CALL CRAWLTO(X,Y+DELTA);
315   4                          DO CASE RUBBER ;
316   5                          ;
317   5                          DO;
318   6                          NY1=NY1+DELTA;NY2=NY2+DELTA;
320   6                          CALL TOGGLELINE(NX1,NY1,NX2,NY2
          -    );
321   6                          END;
322   5                          DO;
323   6                          NY1=NY1+DELTA;  NY2=NY2+DELTA;
325   6                          CALL TOGGLE$RECT(NX1,NY1,NX2,NY
          -    2);
326   6                         END;
327   5                          DO;
328   6                          NY2=NY2+DELTA;
329   6                          CALL TOGGLE$CIRCLE(NX2,NY2,RAD1
          -    US);
330   6                          END;
331   5                          DO;
332   6                          NY2=NY2+DELTA;
333   6                          CALL RUBBER$LINE(NX1,NY1,NX2,NY
          -    2);
334   6                         END;
335   5                          END;
336   4                          END;

337   3        IF (CHAR='D') THEN DO;
339   4                          CALL CRAWLTO(X,Y-DELTA);
340   4                          DO CASE RUBBER ;
341   5                          ;
342   5                          DO;
343   6                          NY1=NY1-DELTA;NY2=NY2-DELTA;
345   6                          CALL TOGGLELINE(NX1,NY1,NX2,NY2
          -    );
346   6                          END;
347   5                          DO;
348   6                          NY1=NY1-DELTA;  NY2=NY2-DELTA;
```

```
350   6                         CALL TOGGLE$RECT(NX1,NY1,NX2,NY2
          -   );
351   6                         END;
352   5                          DO;
353   6                          NY2=NY2-DELTA;
354   6                          CALL TOGGLE$CIRCLE(NX2,NY2,RADI
          -   US);
355   6                          END;
356   5                          DO;
357   6                          NY2=NY2-DELTA;
358   6                          CALL RUBBER$LINE(NX1,NY1,NX2,NY
          -   2);
359   6                          END;
360   5                          END;
361   4                          END;
362   3        IF (CHAR='L') THEN DO;
364   4                          CALL CRAWLTO(X-DELTA,Y);
365   4                          DO CASE RUBBER ;
366   5                          ;
367   5                          DO;
368   6                          NX1=NX1-DELTA;NX2=NX2-DELTA;
370   6                          CALL TOGGLELINE(NX1,NY1,NX2,NY2
          -   );
371   6                          END;
372   5                          DO;
373   6                          NX1=NX1-DELTA;NX2=NX2-DELTA;
375   6                          CALL TOGGLE$RECT(NX1,NY1,NX2,NY
          -   2);
376   6                          END;
377   5                          DO;
378   6                          NX2=NX2-DELTA;
379   6                          CALL TOGGLE$CIRCLE(NX2,NY2,RADI
          -   US);
380   6                          END;
381   5                          DO;
382   6                          NX2=NX2-DELTA;
383   6                          CALL RUBBER$LINE(NX1,NY1,NX2,NY
          -   2);
384   6                          END;
385   5                          END;
386   4                          END;
387   3        IF (CHAR='R') THEN DO;
```

```
389   4                         CALL CRAWLTO(X+DELTA,Y);
390   4                         DO CASE RUBBER ;
391   5                         ;
392   5                         DO;
393   6                         NX1=NX1+DELTA;NX2=NX2+DELTA;
395   6                         CALL TOGGLELINE(NX1,NY1,NX2,NY2
          -    );
396   6                         END;
397   5                         DO;
398   6                         NX1=NX1+DELTA;NX2=NX2+DELTA;
400   6                         CALL TOGGLE$RECT(NX1,NY1,NX2,NY
          -    2);
401   6                         END;
402   5                         DO;
403   6                         NX2=NX2+DELTA;
404   6                         CALL TOGGLE$CIRCLE(NX2,NY2,RADI
          -    US);
405   6                         END;
406   5                         DO;
407   6                         NX2=NX2+DELTA;
408   6                         CALL RUBBER$LINE(NX1,NY1,NX2,NY
          -    2);
409   6                         END;
410   5                         END;
411   4                         END;
412   3       IF (CHAR='A') THEN DO;
414   4                         PX1=X; PY1=Y;
416   4                         END;
417   3       IF (CHAR='B') THEN DO;
419   4                         PX2=X; PY2=Y;
421   4                         CALL DDA(PX1,PY1,PX2,PY2);
422   4                         DO CASE RB ;
423   5                         ;
424   5                         DO ;NX1=PX1;NY1=PY1;NX2=PX2;NY2=
          -    PY2;
429   6                             RUBBER=01;END;
431   5                         ;
432   5                         ;
433   5                         ;
434   5                         END;
435   4                         END;
436   3       IF (CHAR='S') THEN DO;
```

```
438   4                       RB=01;
439   4                       END;
440   3       IF (CHAR='X') THEN DO;
442   4                       RB=00;RUBBER=00;
444   4                       END;
445   3       IF (CHAR='F') THEN DO;
447   4                       RB=02;
448   4                       END;
449   3       IF (CHAR='G') THEN DO;
451   4                       RB=03;
452   4                       END;
453   3       IF (CHAR='H') THEN DO;
455   4                       RB=04;
456   4                       END;
457   3       IF (CHAR='J') THEN DO;
459   4                       PX2=X;PY2=Y;
461   4                       CALL DDA(PX1,PY1,PX2,PY2);
462   4                       DO CASE RB;
463   5                       ;
464   5                       ;
465   5                       ;
466   5                       ;
467   5                       DO;
468   6                        NX1=PX1;NY1=PY1;NX2= PX2;NY2=PY
          -   2;
472   6                        RUBBER=04;
473   6                       END;
474   5                       END;
475   4                       END;
476   3         IF (CHAR='K') THEN DO;
478   4                       PX2=X;  IF PX2>PX1 THEN RADIUS=PX
          -   2-PX1;
481   4                                          ELSE RADIUS=PX1-
          -   PX2;
482   4                       CALL CIRCLE(PX2,PY2,RADIUS);
483   4                       DO CASE RB;
484   5                       ;
485   5                       ;
486   5                       ;
487   5                       DO;
488   6                       NX2=PX2;NY2=PY2;
490   6                       RUBBER=03;
```

```
491   6                     END;
492   5                     ;
493   5                     END;
494   4                     END;
495   3       IF (CHAR='M') THEN DO;
497   4                     PX2=X;PY2=Y;
499   4                     CALL LINERECT(PX1,PY1,PX2,PY2);
500   4                     DO CASE RB;
501   5                     ;
502   5                     ;
503   5                     DO;
504   6                     NX1=PX1;NY1=PY1;NX2=PX2;NY2=PY2;
508   6                     RUBBER=02;
509   6                     END;
510   5                     ;
511   5                     ;
512   5                     END;
513   4                     END;
514   3       END;
515   2       END TURTLE;

                  /* MAIN LEVEL */

516   1       PLOTMODE=0;
517   1       CALL CLEARSCREEN(0000H);
518   1       DO WHILE TRUE;
519   2       CHAR =GET;
520   2       CALL PUT(CHAR);
521   2       IF (CHAR ='L') THEN CALL DDA(WORDIN,WORDIN,WOR
          -     DIN,WORDIN);
523   2       IF (CHAR ='D') THEN CALL CLEARSCREEN (DARK);
525   2       IF (CHAR ='B') THEN CALL CLEARSCREEN (BRIGHT);
527   2       IF (CHAR ='Q') THEN CALL LINERECT(WORDIN,WORDI
          -     N,WORDIN,WORDIN);
529   2       IF (CHAR ='T') THEN CALL TURTLE;
531   2       IF (CHAR ='C') THEN CALL CIRCLE(WORDIN,WORDIN,
          -     WORDIN);
533   2       CALL CRLF;
534   2       END;
535   1       END GRAPH;
```

APPENDIX D

****

MODULE INFORMATION:

```
      CODE AREA SIZE      = 0F3BH   3899D
      VARIABLE AREA SIZE  = 00C8H    200D
      MAXIMUM STACK SIZE  = 000EH     14D
      506 LINES READ
      0 PROGRAM ERRORS
```

END OF PL/M-80 COMPILATION